An Intelligent Offline Handwriting Recognition System Using Evolutionary Neural Learning Algorithm and Rule Based Over Segmented Data Points

Ranadhir Ghosh

School of Information Technology and Mathematical Sciences,
University of Ballarat, Australia. r.ghosh@ballarat.edu.au

Moumita Ghosh

School of Information Technology and Mathematical Sciences,
University of Ballarat, Australia. m.ghosh@ballarat.edu.au

In this paper we propose a novel technique of using a hybrid evolutionary method, which uses a combination of genetic algorithm and matrix based solution methods such as QR factorization. The training of the model is based on a layer based hierarchical structure for the architecture and the weights for the Artificial Neural Network classifier. The architecture for the classifier is found using a binary search type procedure. The hierarchical structured algorithm (EALS-BT) is also a hybrid, because it combines the Genetic Algorithm based method with the Matrix based solution method for finding weights. A heuristic segmentation algorithm is initially used to over segment each word. Then the segmentation points are passed through the rule-based module to discard the incorrect segmentation points and include any missing segmentation points. Following the segmentation the contour is extracted between two correct segmentation points. The contour is passed through the feature extraction module that extracts the angular features, after which the EALS-BT algorithm finds the architecture and the weights for the classifier network. These recognized characters are grouped into words and passed to a variable length lexicon that retrieves words that have the highest confidence value.

ACM Classification: I.4 (Image Processing and Computer Vision)

1. INTRODUCTION

Off-line handwriting recognition is the extended field of Optical Character Recognition (OCR) (LeCun *et al*, 1990). The hand-written messages are scanned into a computer file in two-dimension image representation. That scanned image is passed through the recognition system. Hence there is no interaction with the user. There are a few basic steps in offline handwriting recognition systems (Koerich *et al*, 2003). These are segmentation, feature extraction, and classification.

Segmentation is the first subsystem in offline handwritings recognition systems. The existing research has utilized many different approaches for segmentation in handwriting recognition. Some of them have used ANNs for the segmentation of cursive words (Eastwood *et al*, 1997; Blumenstein and Verma, 1999). Most researchers tend to measure the success of their systems by their findings from the character or word recognition phases. Segmentation plays an important role

Manuscript received: 25 August, 2003
Communicating Editor: Pheobe Chen

in the overall process of handwriting recognition (Lu and Shridhar, 1999). Cursive word segmentation deserves particular attention since it has been acknowledged as the most difficult of all handwriting problems. The structures of handwritten characters follow certain rules. If those are identified and implemented properly then the segmentation could help the word recognizing system to generate good word recognition results. Wrong segmentation can mislead the word recognition system into recognizing the characters incorrectly. In this paper we are using a rule based segmentation system to segment cursive handwritten words.

Different recognizers for handwriting recognition systems exist. One approach is the Hidden Markov model (Bunke *et al*, 1995) in handwriting recognition. The most popular approach is using neural network as a recognizer in offline handwriting recognition. Different learning algorithms have been tried in this process. The most common one is error back propagation algorithm (LeCun *et al*, 1989). The problem with the existing algorithms is that they take a long time to train. Also, no proper system exists that can identify the neural network parameters such as the number of hidden neurons. In this work we are using a hybrid-training algorithm that takes less time to converge than the traditional back propagation algorithm, which is also capable of selecting proper neural network architecture simultaneously to improve the recognition rate.

The main aim of this research is to build a word recognition system for offline handwriting recognition. In this proposed word recognition system, rule based segmentation methods are used for the handwritten words. Following segmentation, the contour between two consecutive segmentation points is extracted. The contour is a sequence of consecutive coordinates of the points. The contour is passed through the feature extraction module that extracts the structural features of the character. After which the EALS-BT algorithm finds the architecture and the weights for the neural network classifier. These recognized characters are grouped together into words and passed to a variable length lexicon that retrieves words that have the highest confidence value.

2. RESEARCH METHODOLOGY

The research methodology can broadly be classified into four modules, such as Segmentation, Feature Extraction, Contour Extraction, EALS-BT classifier, and Neural Network Based Lexicon Analyser. The interconnection of the modules is shown in Figure 1.

2.1 Segmentation

The main objective of the Segmentation step is to segment the cursive word into individual characters. The steps are described in Figure 2. The segmentation follows the following steps:

1. **Step 1:** Compute Baselines.
2. **Step 2:** Over-segment the word.
3. **Step 3:** Pass the segmentation points through the rule base to detect the incorrect segmentation point.
4. **Step 4:** Output the correct segmentation points.

2.1.1 Baseline

Baselines are utilized for many reasons; e.g., normalizing sizes, correcting rotations, and deriving features. In this approach we are computing five lines for segmentation purposes. These are upper baseline, lower baseline, middle baseline, ascender line, and descender line. All the baselines are computed with respect to the horizontal pixel density.

Upper baseline: Upper baseline is the line that goes through the top of the lower case characters (e.g. a, n etc.). The line is described in Figure 3.

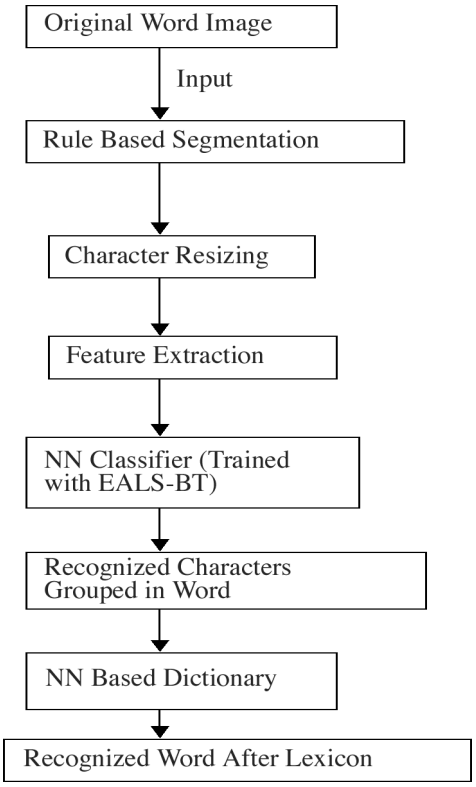

Figure 1: Interconnection of modules within the system

Lower baseline: Lower baseline is the line that goes through the bottom of the lower case characters (e.g. a, n etc.). The line is described in Figure 3.

Middle baseline: The middle baseline corresponds to the writing line on which the word was written. The line is described in Figure 3.

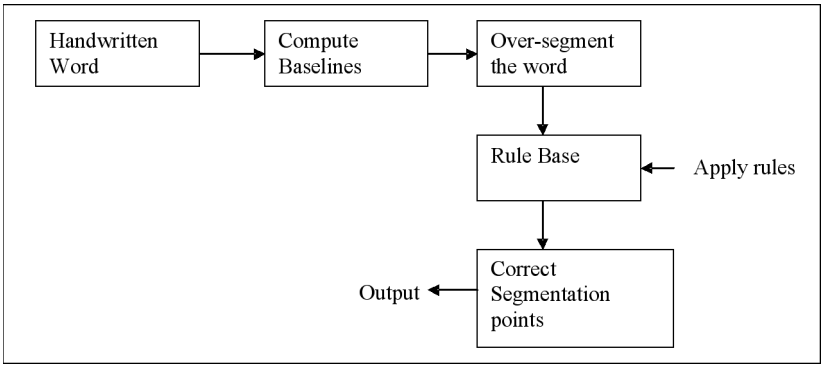

Figure 2: Steps in Segmentation

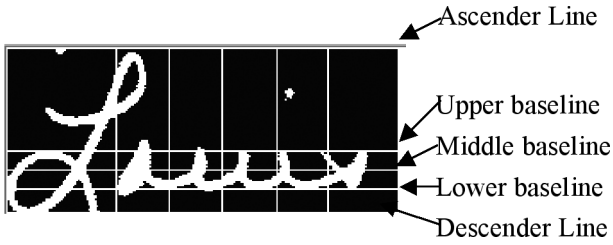

Figure 3: Baselines

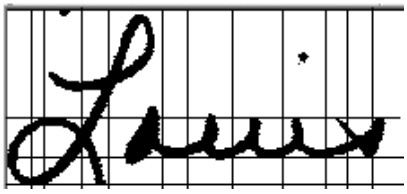

Figure 4: Over segmentation

Ascender line: Ascender line corresponds to the line passes through the topmost point of the word. The line is described in Figure 3.

Descender line: Descender line corresponds to the line passes through the bottommost point of the word. The line is described in Figure 3.

2.1.2 Over segmentation

This module is used to assign a Candidate Segmentation Point (CSP) that could be validated through the rule-based system for further processing. A heuristic over segmentation algorithm is used that incorporates the vertical histogram change. A vertical histogram is drawn at each point in the column and the change in vertical density is noted. Where the change is drastic, the possible Candidate Segmentation Point is drawn. The step is shown in Figure 4.

2.1.3 Rule based validation

The over-segmented word is passed through the rule based validation module where rules are written on the basis of the contour characteristic of a character (such as a loop, a hat shape etc.) described in the following section. According to the rules the incorrect segmentation points are removed from the character.

Rule 1: Detect a loop (closed area) and remove the segmentation points within a loop. Add a segmentation point after end of the loop as a Candidate Segmentation Point (CSP). The rule is described in the Figure 5.

Rule 2: Detect the hat shape and remove the segmentation point within the hat shape contour. Add an extra segmentation point after the end point of the hat shape. The hat shape is described as the '∧' or 'v'. The rule is described in the Figure 6.

Rule 3: Add a few missing segmentation points. The average distance between two segmentation points is calculated taking the average of all segmentation points. If the distances cross the

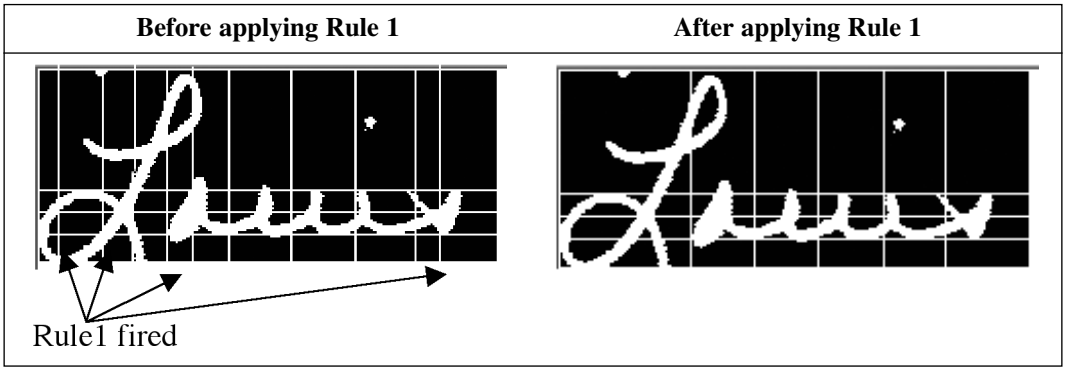

Figure 5: Rule 1

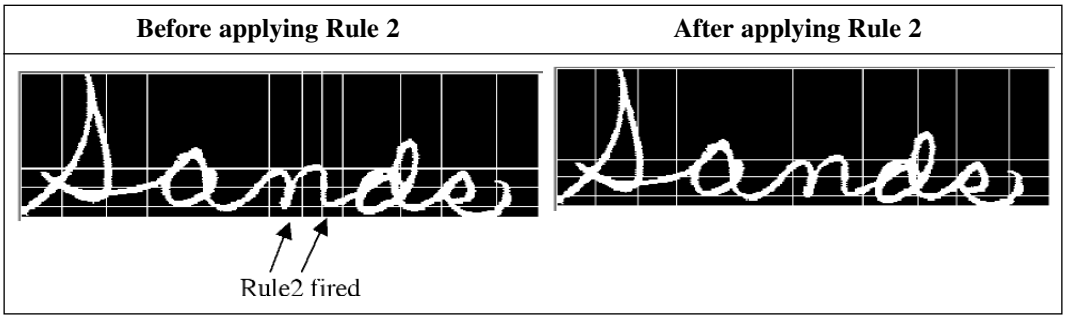

Figure 6: Rule 2

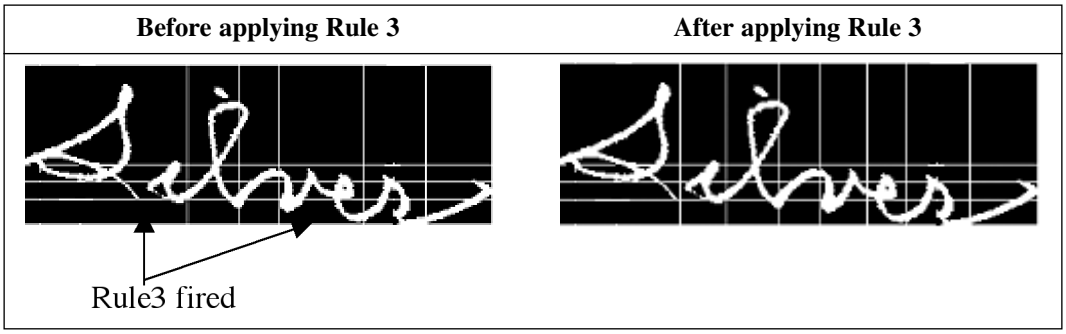

Figure 7: Rule 3

threshold value a Candidate Segmentation Point is added as missing one. The rule is described in the Figure 7.

Rule 4: Delete a few irrelevant segmentation points. The irrelevant points are detected by the average width between two segmentation points. The average distance between two segmentation points is calculated taking the average of all segmentation points. If the distance is less than the average width segmentation point is deleted as an irrelevant one.

2.2. Contour extraction

This section describes the proposed methodology for extracting the contour between two consecutive segmentation points. The contour between two consecutive segmentation points is extracted using the following few steps.

Step 1: The pixels at the first segmentation point are disconnected.
Step 2: The pixels at the second segmentation point are disconnected.
Step 3: The closest distance from the first black pixel at first segmentation point to the three base lines is determined.
Step 4: Follow the contour path across that baseline having minimum distance to find the connecting contour.
Step 5: Mark it as visited once it is visited. If the contour is already visited then that path is discarded and other part is taken, if one exits.

2.3. Feature extraction

A novel feature extraction technique is used to extract the feature between the two contours. The feature extracted in this methodology is a structural feature.

2.3.1.Slope

The slope of the consecutive points is estimated using linear regression. The rate of change of the slope is used as the main feature. The feature is described in Figure 8.

The input to the feature extraction module is the set of coordinate (x, y) of the contour extracted during the contour extraction phase. With the coordinates the slope of two consecutive points are estimated. The slope estimation is done using linear probing. Linear regression attempts to explain this relationship with a straight line fit to the data. The linear regression model postulates that

$$Y = a + bX \tag{1}$$

The coefficients a and b are determined by the following equations.

$$b = \frac{2\left(x_1 y_1 + x_2 y_2\right) - \left(x_1 + x_2\right)\left(y_1 + y_2\right)}{2\left(x_1^2 + x_2^2\right) - \left(x_1 + x_2\right)^2} \tag{2}$$

$$a = \frac{\left(y_1 + y_2\right) - b\left(x_1 + x_2\right)}{2} \tag{3}$$

The slope between the two point (x1, y1) and (x2, y2) is represented by the parameter b.

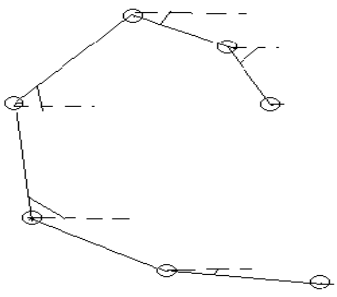

Figure 8: Direction Feature

2.3.2.Slope Direction Change (Up/Down)

The point with respect to the main body of the contour where the direction is changing is also taken care of. The change of direction is denoted by the contour that is changing direction upwards to downward or vice versa.

2.3.3.Stroke length

Stroke length is a local feature. It represents the length of the line joining two consecutive points. The length is normalized with respect to the total length of the body. The length is calculated using the Euclidian distance between the two points.

2.3.4. Linearity

Linearity is a neighbourhood feature. The average square distance between every two consecutive points in the vicinity of (x (t), y(t)) and the straight line joining the first and last position in the vicinity is called Linearity. Linearity LN (t) can be defined as follows:

$$LN(t) = \frac{1}{N}\sum_{i} d_i^2 \tag{4}$$

2.3.5. Angle of vicinity

The slope of the straight line joining the first and the last point in the vicinity of (x (t), y (t)) is described by the cosine of its angle.

2.3.6. Number of ascenders

Number of ascenders is a global feature. It denotes the number of points above the base line.

2.3.7. Number of descenders

Number of descenders is a global feature. It denotes the number of points below the base line.

2.3.8. Vertical position of start/end point

Vertical position is a global feature. It represents the vertical position of the start and the end points with respect to the base line.

2.3.9. Curliness

Curliness is a neighbourhood feature. Six consecutive neighbours are considered. The neighbours in either direction are considered, i.e. three in a forward direction and three in a backward direction. Curliness C(t) is the feature that describes the deviation from the straight line in the vicinity of (x(t), y(t)). The feature is descried in Figure 9. It is based on the ratio of the length of the trajectory and the maximum side of the boundary box:

$$C(t) = \frac{\sum_{i} l_i}{\max(\Delta x, \Delta y)} \tag{5}$$

Where l_i denotes the length the line segment joining two consecutive points.

2.4 EALS-BT classifier model

A hybrid evolutionary technique is used to find the neural network architecture and weights. The details of this algorithm are described in (Ghosh, 2003). The technique is a combination of genetic

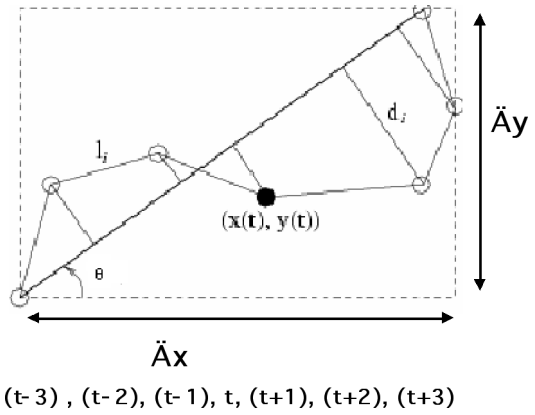

Figure 9: Curliness Feature

algorithm and matrix based solution methods such as QR factorization. The training of the model is based on a hierarchical structure, which is evolved with its own architecture and weights using a method called Evolutionary Algorithm and Least Square to find its weights and a type of Binary Search for the architecture (EALS-BT). In Figure 10, the flowchart for the dynamics of the combination methodology for searching the architecture and weights is given. The two separate modules for architecture and weights are referred as findArchitecture and EALS module respectively. A simple rule base can describe the working of the stopping criterion for the algorithm.

Rule 1: If the current output is satisfactory, then stop the algorithm, else check rule 2.

Rule 2: If the stopping criterion for the weight search is met and the search is completely exhausted (in terms of number of iterations) then stop else check rule 3.

Rule 3: If the stopping criterion for the weight search is met then go to rule 4 else go to the next generation for the EALS.

Rule 4: If the stopping criterion for EALS is met then go to rule 1, else initialize for the next number of hidden neurons for EALS.

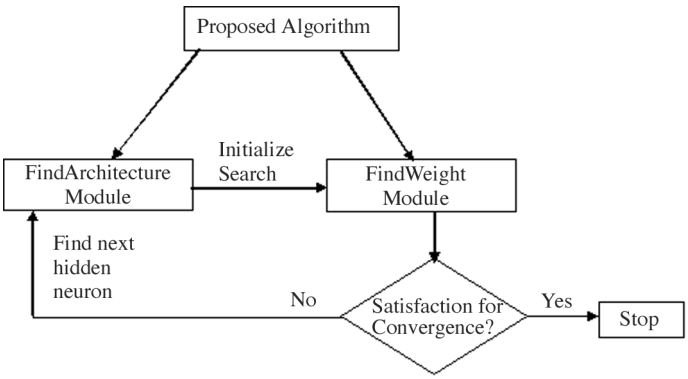

Figure 10: Hierarchical structures for weight and architecture modules

2.4.1 Issue of combining the GA and least square method

How we can combine the evolutionary algorithm with the least square method is again a very important issue as there are many possibilities of joining the two independent modules. The LS method is called after the convergence of the GA is completed. After a certain number of generations for the GA, the best fit population is halved and the lower half is used as the weights for the hidden layer and those weights are used for the LS method. The flowchart in Figure 11 shows the input/output relationship between the two modules **findArchitecture** and **EALS**. The decision box shows the combination of all the rules described in the beginning of the sections.

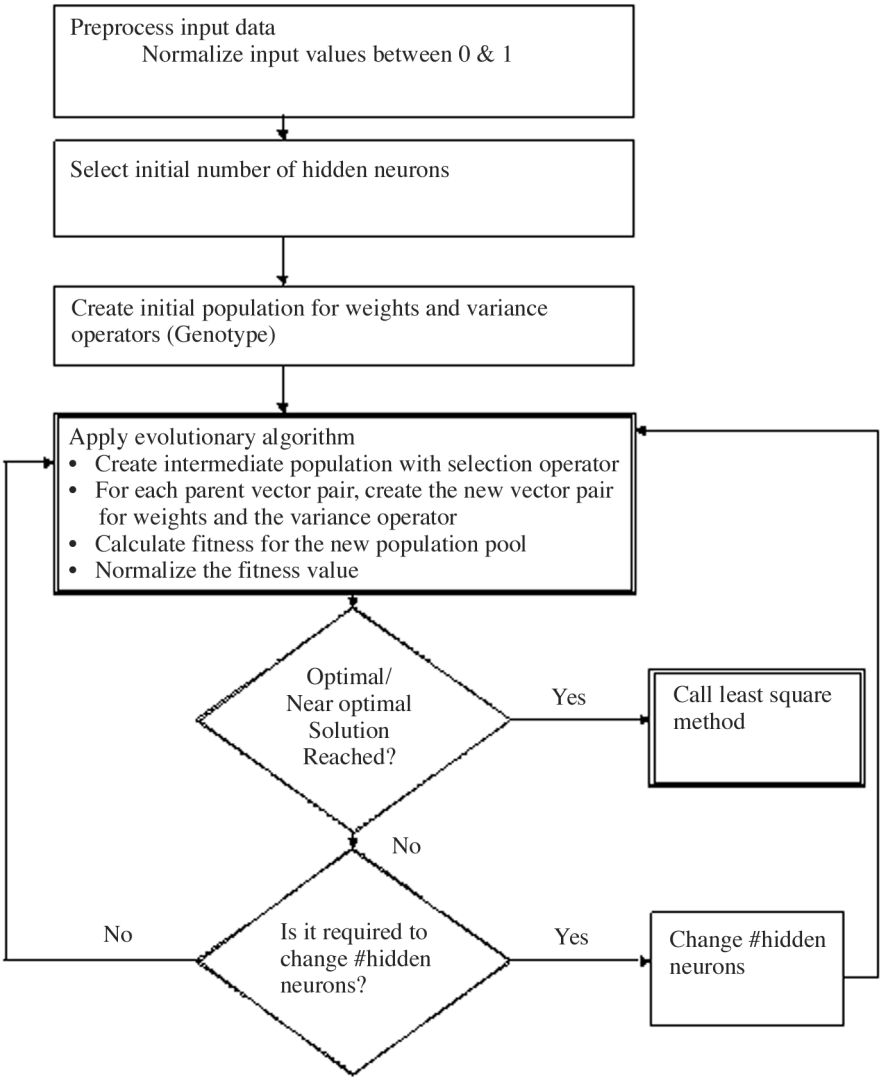

Figure 11: Flowchart details for EALS and its Input/Output combination with the weight search

2.4.2 Stopping criteria for EALS

The Stopping criterion for EALS is also based on a few simple rules. All the rules are based on the current train and test output and the maximum number of generations for the evolutionary algorithm. In the following, we describe the stopping criterion for the convergence of the evolutionary algorithm.

If (best_RMS_error[1] < goal_RMS_error) then Stop

Else if (number_of_generation = total_number_of_generation[2]) then
Stop

Else if (train_classification_error is increased in #m[3] consecutive generation) then
Stop

Else continue.

2.4.3 FindArchitecture Module

We used a binary search type for EALS (EALS-BT) to find the number of hidden neurons. We use a binary search type to find the optimal number. A pseudo-code of the algorithm is given below:

Step 1: Find the % test classification error & train_classification_error (error_min) for #min number of hidden neurons :

error_min = (train_classification_error (%)+ test_classification_error (%)) / 2

Step 2: find the % test classification error & train classification error (error_max) for #max number of hidden neurons

error_max = (train_classification_error (%) + test_classification_error (%)) / 2

Step 3: Find the % test classification error & train classification error (error_mid) for #mid (mid = (min + max) / 2) number of hidden neurons

error_mid = (train_classification_error (%) + test_classification_error (%)) / 2

Step 4: if (error_mid < error_min) and (error_mid > error_max) then
min = mid
mid = (min + max / 2)
else
max = mid
mid = (min + max / 2)
end if

Step 5: Go to Step1, if (mid > min) and (mid < max) Else go to Step 6

Step 6: #number of hidden neurons = mid.

2.5 Lexicon analyzer model

A neural network based dictionary was used for recognizing words following the recognition of individual characters. The normalized ASCII value for each character was taken as a feature for the neural network. The Hamming neural network is trained with the words in the dictionary. The network fires the output neuron that matches the word in the dictionary. Figure 12 shows the representation of lexicon analyzer.

[1] The best_RMS_error is the best of the RMS error from the population pool
[2] Total number of generations is considered as 30
[3] m is considered as 3

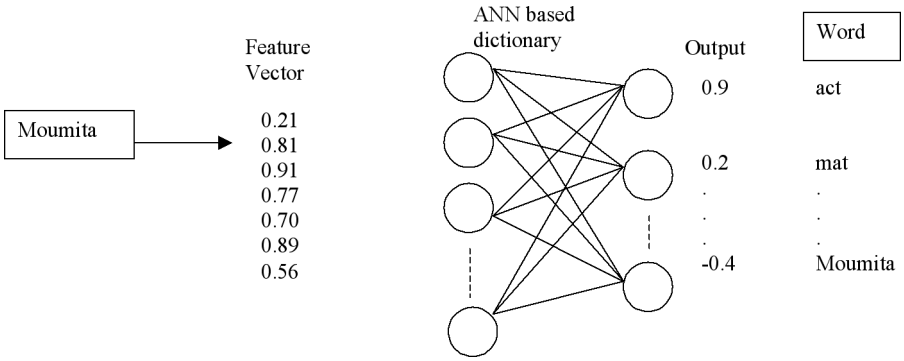

Figure 12: ANN based lexicon analyzer

3. IMPLEMENTATION AND EXPERIMENTAL RESULTS

In this paper, a subset of the CEDAR dataset is used. A number of experiments were conducted. Samples of handwritten words from the CEDAR benchmark dataset were used to test the segmentation module. The character dataset of the CEDAR dataset was also used to train the neural network classifier. All the algorithms were implemented in C++ on a UNIX platform. The number of characters used for training and testing respectively was 7000 and 2000 respectively. There were 26 outputs representing the uppercase characters (A-Z) and 26 representing the lowercase characters (a-z).

3.1 Segmentation results

To test the accuracy of the rule based novel segmentation algorithm three criteria were used for the segmentation results. These are 1) number of over segmentations, 2) missed segmentations and 3) bad segmentations. Over segmentation is denoted when more than two segmentation lines segment the character. Missed segmentation is denoted when a correct segmentation point is missing between two characters. Bad segmentation denotes the segmentation point that does not separate two characters properly. The error rates are shown in Table 1.

3.2 Character recognition results

The results obtained for that character recognition are shown in Table 2. The results are compared with the traditional back propagation algorithm.

The results in Table 2 show a comparative study on the improvement of EALS-BT algorithm over the traditional back propagation algorithm for the CEDAR dataset.

3.3 Word recognition results

The word recognition results are shown in Table 3. The words are passed through the lexicon analyser. The word recognition results we got before passing through the lexicon analyser was 75%. The recognition rate after passing through the lexicon was 96%.

Over segmentation (%)	Missed (%)	Bad (%)
20.02	0.2	8.7

Table 1: Segmentation Results

Lower/Upper case	Dataset	Recognition Rate with Back propagation Algorithm [%]	Recognition Rate with EALS-BT Algorithm [%]
Lower Case	Training	100	99.8
Lower Case	Testing	87.3	92.4
Upper case	Training	99.7	100
Upper case	Testing	86.4	95.6

Table 2: Character Recognition Results

Length of lexicon	Word Recognition Result	
	Before passing through the Lexicon analyzer [%]	After passing through the Lexicon analyzer [%]
50	75	96
100	75	95

Table 3: Word Recognition Results

4. ANALYSIS

In this section the analysis of all the experimental results conducted are given with a brief discussion at the end of every analysis. The entire analysis can be broadly divided into two sections. In the first section, convergence and other properties of the proposed approach are analysed in detail. In the second part, analysis and discussions are given with comparison to the traditional back propagation algorithm.

4.1 Analysis of convergence property for the EALS-BT approach

In this section the convergence of the filter approach is analysed with the following two basic convergence properties –

1. How the training classification error decreases over time (Is there a steady decrease of training classification error).
2. How the test classification error decreases over the training time.

The following figure (Figure 13) shows the classification error during training over the generation of the two datasets. Figure 13 shows that classification error decreased steadily over the

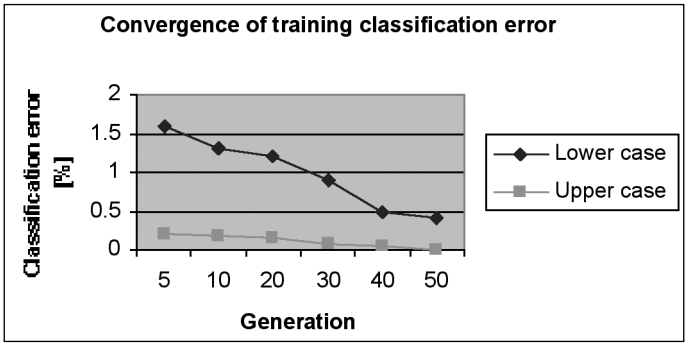

Figure 13: Convergence of training classification error

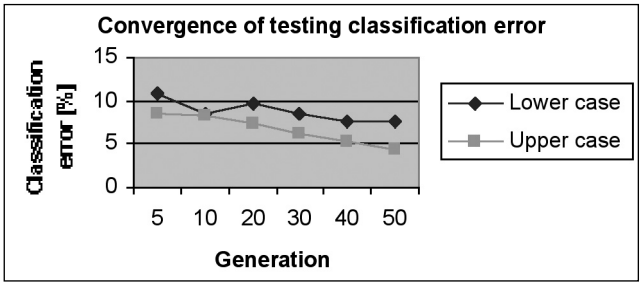

Figure 14: Convergence of test classification error

generation of the lower case dataset. But there was a fluctuation in case of the upper case dataset. For the lower case dataset the classification error varied in the range 1.6% to 0.2%, and for the upper case dataset the classification error varied in the range of 0.2% to 0%.

Figure 14 shows the classification error during testing over the generation of the two datasets. Figure 14 shows that for the upper case dataset the classification error varied in the range of 8.4% to 4.4%, but for the lower case dataset the classification error varied in the range of 10% to 7.6%.

4.2 Performance analysis for lower case and uppercase classifier on UNIPEN dataset

The experiment was started with two neural classifiers for lowercase and uppercase. The upper case characters achieved a higher classification result than the than lowercase characters. That is due to the increasing ambiguity of lower case characters. The shapes of the uppercase characters are more straightforward and unambiguous, but lowercase characters are very ambiguous. Several characters in lowercase like (i, l, j), and (c, e) sometimes follow similar shapes and thus caused miss-recognition a number of times.

4.3 Comparison with Back propagation algorithm

In this section, analyses are given for the performance of the proposed approaches compared with the traditional back propagation algorithm. The following two sub sections represent the comparison with respect to the classification accuracy and the comparison with respect to the time complexity of the proposed approach compared to the back propagation algorithm.

4.3.1 Analysis of classification accuracy

This section analyses the classification accuracy for the three approaches. The following figure (Figure 15) shows a comparison study of the classification accuracy.

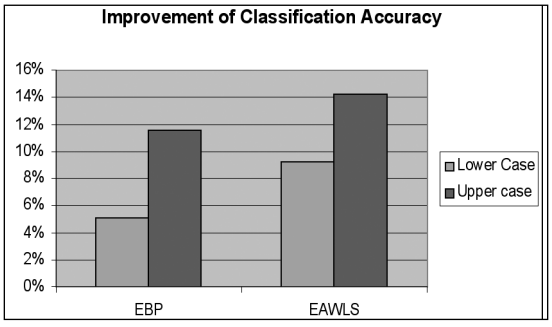

Figure 15: Improvement of Classification accuracy

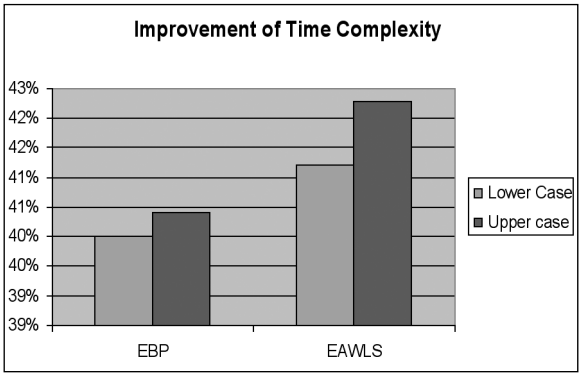

Figure 16: Time complexity for different approaches

4.3.2 Analysis of time complexity

This section analyses the time complexity for the three approaches. Figure 16 shows a comparison study of the time complexity with back propagation algorithm.

5. CONCLUSIONS

In this paper a novel hybrid evolutionary technique is used for offline handwriting recognition, which uses a combination of genetic algorithm and matrix based solution methods such as QR factorization. The technique produces satisfactory results. The results further improve with application of the Hamming neural network as a lexicon analyzer. However the recognition algorithm takes considerable memory for the matrix operations. In future we will try to improve the memory complexity of the system.

REFERENCES

BLUMENSTEIN, M., and VERMA, B. (1999): A new segmentation algorithm for handwritten word recognition. *Proc. IJCNN International Joint Conference on Neural Networks*, Washington DC, USA, 872–882.

BUNKE, H., ROTH, M., and SCHUKAT-TALAMAZZINI, E.G. (1995): Offline cursive handwriting recognition using hidden Markov models. *Proc. Pattern Recognition*, 28(9): 1399–1413.

EASTWOOD, B., JENNINGS, A., and HARVEY, A. (1997): A feature based neural network segmenter for handwritten words. *Proc. ICCIMA International Conference on Computational Intelligence and Multimedia Applications*, Gold Coast, Australia, 286–290.

GHOSH, R. and VERMA, B. (2003): Finding architecture and weights for ANN using evolutionary based least square algorithm, *Proc. IJNS International Journal on Neural Systems*, 13(1): 13–24.

KOERICH, A.L., SOBOURIN, R., and SUEN, C.Y. (2003): Large vocabulary off-line handwriting recognition: A survey. *Proc. PAA Pattern Analysis Application*, 6(2): 97–121.

LECUN, Y., BOSER, B., DENKER, J.S., HENDERSEON, D., HOWARD, R.E., HUBBARD, W. and JACKEL, L.D. (1990): Handwritten digit recognition with back-propagation network. *Proc. ANIPS Advance in Neural Information Processing System*, San Mateo, CA, 598–605.

LU, Y., and SHRIDHAR, M. (1988): Character segmentation in handwritten word – An overview. *Proc. Pattern Recognition*, 77–96.

BIOGRAPHICAL NOTES

Ranadhir Ghosh

After completing his BSc in computer science and engineering from Bangalore University, India, Ranadhir Ghosh obtained his MSc in IT from Bond University, and later obtained a PhD from Griffith University in 2003 with an academic excellence award. He is currently a lecturer in the school of ITMS at University of Ballarat. His expertise is in evolutionary neural learning and its applications. He has many publications in various international journals, conferences, and book chapters.

Moumita Ghosh

Momita Ghosh has a BSc in computer science and technology from Bengal Engineering College, Shibpur, India. She completed an honours degree in IT at Griffith University in 2003, and is currently a PhD student working in hybrid optimisation areas in University of Ballarat. Her research interests include optimisation methods, neural networks, and genetic algorithms.