International Journal of Soft Computing 3 (5): 373-381, 2008
ISSN: 1816-9503

Structure Analysis of Multilayer Perceptron Network for Handwritten Tamil Character Recognition Using Levenberg-Marquardt Algorithm

[1]J. Sutha and [2]N. Ramaraj
[1]Department of Computer Science and Engineering, Sethu Institute of Technology, Kariapatti
[2]G.K.M. College of Engineering and Technology, Chennaz

Abstract: This study addresses the problem of recognizing handwritten Tamil characters, the popular south Indian Language. Applying neural network for recognizing a large volume of high dimensional data is a difficult task as the training process is computational expensive. Back propagation(BPN) training algorithm has been applied to various problems by researchers. However, the main drawback in applying that algorithm for real world problem is the low convergence speed in the entire learning process. This proposed system uses Levenberg-Marquardt (LM) algorithm for training a multilayer perceptron (MLP) network with one hidden layer. It is a Hebbian-based algorithm then the training process converges quickly compared to BPN algorithm. Various preprocessing algorithms and feature extraction techniques performed prior to the recognition of handwritten Tamil characters are also implemented. The performance of the LM algorithm has been checked for the recognition of handwritten Tamil characters and structure analysis is carried out to find the optimum structure of the MLP network. Test results indicate that the proposed system provide good recognition accuracy of 94.1% for handwritten Tamil characters and structure analysis suggest that the appropriate structure should have 50 hidden nodes with tangent sigmoid function for hidden nodes, pure linear function for output nodes with a maximum training epoch of 150 to achieve the higher recognition rate for the MLP network trained using LM algorithm.

Key words: Handwritten character recognition, preprocessing, segmentation, feature extraction, fourier descriptors, transition values, average pixel values, distances and angle features, LM algorithm

INTRODUCTION

Today, character recognition is an active research field. Recognition of handwritten characters was still remains a difficult problem due to the large degree of variability of writing styles, different size and orientation angle of the characters. Recognition of Indian languages are much difficult than English and Numerals because some of the characters in Indian languages are in similar shape. To the best of my knowledge, little work has been done in the area of Tamil character recognition compared with those for English, Numerals, Chinese and Japanese text. Many systems has been developed for the recognition of English language (Blumenstien and Verma, 1998; Bozinovic and Srihari, 1989; Hanmandlu *et al*., 1999), Chinese language (Deng *et al*., 1994) and handwritten numerals (Chung and Wong, 1997; Ha and Bunk, 1997; Lee, 1996). However, less attention had been given to Indian language recognition. Some works have been reported for Devanagari (Bansal and Sinha, 1999), Hindi numerals (Hanmandlu and Ramana Murthy, 2005) and Printed Telugu text (Vasantha and Patvardhan, 2003). The main difficulty of recognition of Tamil handwritten characters is the similarity of more characters in shape.

Several approaches have been attempted for handwritten character recognition in the last years. In this context, algorithms based on neural networks (Blumenstien and Verma, 1997; Chung and Wong, 1997; Deng *et al*., 1994; Hanmandlu *et al*., 1999; Kim and Nam, 1995; Lee, 1996) have been proved to give better results than conventional methods. Although, the Error Backpropagation algorithm has been a significant milestone in neural network research area, it has give low convergence rate. Many attempts have been made to speed up the backpropagation algorithm. Commonly known heuristic approaches like momentum, variable learning rates were only give slight improvement. Several approaches for fast training of multilayer perceptron are discussed in Azimi-Sadjadi *et al*. (1990), Verma (1997) and Park *et al*. (1992).

This study exploits the use of neural networks for off-line Tamil handwritten character recognition. There have

Corresponding Author: J. Sutha, Department of Computer Science and Engineering, Sethu Institute of Technology, Kariapatti

been only a few attempts in the past for the recognition of handwritten Tamil Characters. Some works have been reported for Tamil scripts in Apana *et al*. (2004), Chinnuswamy and Krishnamoorthly (1980), Hewavitharana and Fernando (2002), Joshi *et al*. (2004) and Suresh *et al*. (1999). Although, most of these handwriting recognition applications concentrate on on-line handwriting (Aparna *et al*., 2004; Joshi *et al*., 2004), handprinted Tamil characters (Chinnuswamy and Krishnamoorthly, 1980) there have been attempts on offline Tamil handwritten characters (Hewavitharana and Fernando, 2002; Suresh *et al*., 1999).

TAMIL LANGUAGE

Tamil, which is a south Indian language, is one of the oldest languages in the world. Tamil script is used to write Tamil language in Tamil Nadu, SriLanka, Singapore and parts of Malaysia, as well as to write minority languages such as Badaga. The Tamil alphabet consists of 12 vowels, 18 consonants and one special character (AK). Vowels and consonants are combined to form composite letters, making a total of 247 different characters and some Sanskrit characters. In character recognition point of view, only 123 symbols have to be identified to recognize all 247 characters. The complete Tamil alphabet and composite character formations are given in (Chinnuswamy and Krishnamoorthly, 1980). In this Tamil character recognition process, the most difficult problem is the similar shape of few characters.

PREPROCESSING

The handwritten character data samples were acquired from various students and faculty members both male and female. Their handwriting was sampled on A4 size paper. They were scanned using flat-bed scanner at a resolution of 100 dpi and stored as 8 bit grey scale images. Preprocessing is concerned mainly with the reduction of noise. It is necessary to perform several preprocessing operations prior to the recognition of handwritten scanned documents to remove noise due to erratic hand movements and inaccuracies in digitization of the actual input. Some of the common operations performed prior to recognition are: Smoothing, Thresholding, line segmentation and character segmentation. Once the characters have been segmented, the character image needs to be normalized so that the characters will be the same size and in the same position. Sample handwritten Tamil character image is shown in Fig. 1.

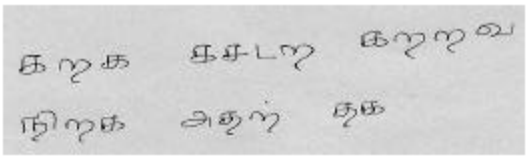

Fig. 1: Sample Handwritten character Image

Smoothing: The task of smoothing is to remove unnecessary noise present in the image. Spatial filters could be used. To reduce the effect of noise, the image is smoothed using a Gaussian filter.

Thresholding: The task of thresholding is to extract the foreground from the background. A number of thresholding techniques have been previously proposed using global and local techniques. Global methods apply one threshold to the entire image while local thresholding methods apply different threshold values to different regions of the image. The histogram of gray scale values of a handwritten character image typically consists of two peaks: a high peak corresponding to the white background and a smaller peak corresponding to the foreground. So the task of determining the threshold gray-scale value is one of determining as optimal value in the valley between the two peaks. Otsu's method of histogram based global thresholding algorithm is used (Shanthi and Duraiswamy, 2005).

Segmentation: Segmentation is a necessary step in order to isolate the text image to be passed to the recognition stage. Segmentation of handwritten text into lines and characters has many sophisticated approaches. This is in contrast to the task of segmenting lines of text into characters, which is straight-forward for machine-printed documents. It can be accomplished by examining the horizontal histogram profile at a small range of skew angles. The task is more difficult in the handwritten domain. Here, lines of text might be undulate up and down and ascenders and descenders frequently intersect characters of neigh-boring lines.

Line segmentation: An input image consists of a uniform text area with distinct text lines. In the line segmentation process, this is broken into constituent text lines. Normally, lines are separated by horizontal gaps and hence the horizontal projection profile is used to identify the segmentation points. Based on the segmentation points the image is segmented into sub images of each line of text.

Character Segmentation: Line segmentation is usually followed by a procedure that separates the text line into characters. Normally, characters are separated by vertical gaps and hence the vertical projection profile is used to identify the segmentation points. Based on the segmentation points the image is segmented into individual character images.

Size Normalization: Once the characters have been segmented, the character image needs to be normalized so that the characters will be the same size and in the same position. People have great variability in their handwriting. Some write small characters and others write larger characters. Scaling reduces or enlarges the size of the characters to a predefined size. All the handwritten characters are normalized into standard size depending on the Aspect Ratio (AR) of the character. Aspect ratio is the ratio of height to width of the image. Irrespective of the size of a handwritten character, all the characters are normalized into Aspect ratio of 1.25. The character images are fed into a feature extraction program for extracting features for each character image.

FEATURE EXTRACTION

Feature extraction plays important role in the success of handwritten character recognition system. In Hanmandlu *et al*. (1999), feature extraction is done using three different approaches, namely, ring, sector and hybrid. The features consist of normalized vector distances and angles are proposed in Hanmandlu and Ramana Murty (2005). In Blumenstien and Verma (1998), the direction features and transition features were used to classify the unknown characters. Fourier features are used to identify the objects in Kim and Nam (1995). In Suresh *et al*. (1999), the normalized feature vector based on 16 directions is used to classify the Tamil handwritten characters. This study proposes 6 features; all are novel features which are used to identify the handwritten Tamil characters.

Boundary tracing: The purpose of the algorithm is to extract the information of the boundary of a handwritten character. Various boundary tracing methods are available. The "eight-neighbor" adjacent method is adopted in this system. The algorithm scans the binary image until it finds the boundary. The searching will follow according to the clockwise direction. For any foreground pixel p, the set of all foreground pixels connected to it is called connected component containing p. The pixel p and its 8-neighbors are shown in Fig. 2.

P_1	P_2	P_3
P_4	P	P_5
P_6	P_7	P_8

Fig. 2: Pixel P and its 8-neighbors

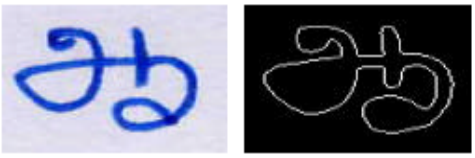

Fig. 3: Boundary tracing of a character

Once a white pixel is detected, it will check another new white pixel and so on. The tracing will follow the boundary automatically. When the first pixel is found, the program will be assigned the coordinates of that position to indicate that this is an origin of the boundary. The tracer will search for the next nearby white pixels. The new found pixel will be assigned as a new reference point and starts the eight-neighbor searching. In this way, the coordinates of the initial point are varied according to the position. As the tracer moves along the boundary of the image, the corresponding coordinates will be stored in an array for the computation of Fourier Descriptors. During the boundary tracing process, the program will always check the condition whether the first coordinates of the boundary are equal to the last coordinates. Once it is obtained means that the whole boundary has been traced and completed the boundary tracing process. Figure 3 shows the boundary tracing result.

Fourier descriptors: Once a boundary image is obtained then Fourier descriptors are found. Boundary tracing of each character image yielded the Cartesian coordinates x(m), y(m), where m = 1,2,...L. of the boundary elements. Since the boundaries are closed curves, it is observed that

$$x(L) = x(1); y(L) = y(1) \quad (1)$$

The fourier analysis involves finding the discrete Fourier transform of the data, x(m), y(m), $1 \le m \le L$, thus obtain the Fourier Coefficients a(k) and b(k) for $1 \le k \le L$, where L is the total number of boundary points found, by applying Eq. (2) and (3).

$$a(k) = \frac{1}{L}\sum_{m=1}^{L} x(m)e^{-jk(2\pi/L)m} \quad (2)$$

$$b(k) = \frac{1}{L}\sum_{m=1}^{L} y(m)e^{-jk(2\pi/L)m} \quad (3)$$

where, x[m] and y[m] are the x and y co-ordinates, respectively of the m^{th} boundary point. The Fourier coefficients derived according to Eq. (2) and (3) are not rotation or shift invariant. In order to derive a set of Fourier descriptors that have the invariant property with respect to rotation and shift, the following operations are defined. For each k compute a set of invariant descriptors r(k).

$$r(k) = \sqrt{|a(k)|^2 + |b(k)|^2} \quad (4)$$

It is easy to show that r(k) is invariant to rotation or shift. A further refinement in the derivation of the descriptors is realized if dependence of r(k) on the size of the character is eliminated by computing a new set of descriptors s(k) as:

$$s(k) = \frac{r(k)}{r(1)} \quad (5)$$

The Fourier coefficients a(k), b(k) and the invariant descriptors s(k), k = 1, 2 ..... L were derived for all of the character specimens.

Transition values: This technique makes use of the topological properties of the characters to identify individual characters which cannot be performed by Fourier descriptors due to its rotational invariance. The purpose of the algorithm is to extract the information of the foreground-to-background and background-to-foreground transition values of boundary of each character in both vertical and horizontal directions. It assumes that all the characters are oriented vertically. Each character is partitioned into 4 equal sections and scans each section in both horizontal and vertical direction to count the number of occurrences of zero-to-one and one-to-zero transition values.

The algorithm for the calculation of transition values is described as follows:

- Partition each character image into four equal sections.
- Count the number of zero-to-one and one-to-zero transition in each section by performing row and column scan.
- Find the average transition value for each scan.

Table 1: Direction of scan for each section

Section	Transition	Direction
1	Horizontal	Left to right
	Vertical	Top to bottom
2	Horizontal	Left to right
	Vertical	Bottom to top
3	Horizontal	Right to left
	Vertical	Top to bottom
4	Horizontal	Right to left
	Vertical	Bottom to top

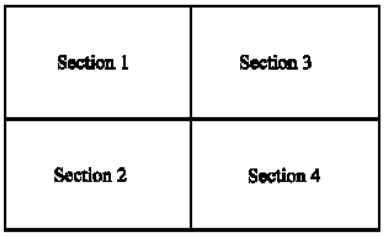

Fig. 4: Section partition of each character

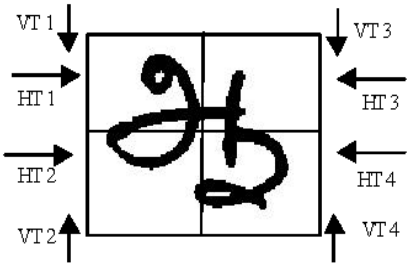

Fig. 5: Transition value calculation of a character

Figure 4 shows the section partition of each character for boundary transition. The direction of scan for each section is shown in Table 1.

Figure 5 shows the transition value calculation of a character.

To determine horizontal transition value for section 1 by counting zero-to-one and one-to-zero transitions as the rows are scanned left to right sequentially and find the average transition value for each scan. The vertical transition value for section 1 is determined by performing column scan instead of a row scan. The average transition value for both vertical and horizontal directions for each section will be calculated. This gives eight boundary transition values for each character. This feature can be very useful to identify the characters with very close Fourier Descriptors because of its rotational invariance.

Average pixel values: This feature is very much useful to avoid the misclassification of characters due to similar shape. The purpose of the algorithm is to find the average foreground pixel values of handwritten character image.

Each character is partitioned into 16 equal sections and scans each section to count the number of occurrences of zero pixels. The average zero pixel values for each section will be calculated. This gives sixteen average pixel values for each character. This feature can be very much useful to identify the characters with similar shape. The algorithm for the calculation of average pixel values is described as follows:

- Partition each character image into sixteen equal sections.
- Scan each section horizontally to count the number of occurrences of zero pixels.
- Find the average value for each scan.

Tenure features: This feature is also very much useful to avoid the misclassification of characters due to similar shape. The purpose of the algorithm is to find the ratio of foreground pixel value in each section with respect to overall foreground pixel values of handwritten character image. The algorithm for the calculation of tenure feature values is described as follows:

- Partition each character image into sixteen equal sections.
- Scan each section horizontally to count the number of occurrences of zero pixels in the section.
- Find the tenure feature value by dividing number of occurrences of zero pixels in each section by the number of occurrences of zero pixels in an image.

Distance and angle features: Normalized vector distance and angle features are extracted from the character image for achieving high recognition rates. The features are more robust as they do not depend on the centroid. In this method, the character image is partitioned into a fixed number of sections. Once the character is bifurcated into sections, the portions lying in each section is used for the extraction of features. It is noted that a character may not present in all the sections. Here, we consider geometric features consisting of vector distances and angles.

Extraction of distance and angle features: The normalized distances and angles are calculated for all the sections. Let n_s be number of '0' pixels present in a section s, with s = 1, 2,...., 16. For each section, the normalized vector distance, which is the sum of distances of all '0' pixels in a section divided by the number of '0' pixels present in that section is calculated by:

$$d_s = \frac{1}{n_s}\sum_{i=1}^{n_s}\sqrt{(x_m - x_i)^2 + (y_n - y_i)^2} \qquad (6)$$

Fig. 6: Section partition of a character 'oo' for the calculation of average pixel values, Tenure features and distance and angle features

where, (x_i, y_i) are the co-ordinates of a pixel in a section and (x_m, y_n) are the co-ordinates of the center of the character image. This gives one normalized vector distance for each section totally gets 16 values for each character. Next, for each section the corresponding angles of pixels are also calculated. The normalized angle, α_s which is taken as another set of features, is calculated as:

$$a_s = \frac{1}{n_s}\sum_{i=1}^{n_s}\tan^{-1}\left[\frac{y_n - y_i}{x_m - x_i}\right] \qquad (7)$$

Both vector distance d_s and vector angles α_s constitute 32 features from 16 sections.

Figure 6 shows section partition of a character 'oo' for the calculation of average pixel values, Tenure features and Distance and angle features.

RECOGNITION

Recognition of handwritten characters is a very complex problem. The characters could be written in different size, orientation, thickness, format and dimension. These will give infinity variations. The capability of neural network to generalize and be insensitive to the missing data would be very beneficial in recognizing handwritten characters. The proposed Tamil handwritten character recognition system uses a neural network based approach to recognize the characters represented by various efficient features. Feed forward Multi Layered Perceptron (MLP) network with one hidden layer trained using Levenberg-Marquardt algorithm has been used to recognize handwritten Tamil characters.

Levenberg marquardt algorithm: The proposed system uses Levenberg-Marquardt algorithm for training a MLP network with one hidden layer. It is a Hebbian-based algorithm then the training process converges quickly compared to BPN algorithm.

In LM algorithm, mean square error function is calculated as:

$$e(x) = \frac{1}{2}\sum_{k=1}^{P}\sum_{l=1}^{m}(t_{kl} - y_{kl})^2 \quad (8)$$

where, y_{kl} is the actual output at the output neuron l for the input k and d_{kl} is the desired output at the output neuron l for the input k. P is the total number of training patterns and m represents the total number of neurons in the output layer of the network. X represents the weights and biases of the network.

The steps involved in training a MLP neural network in batch mode using the Levenberg-Marquardt algorithm are as follows:

- Present all inputs to the network and compute the corresponding network outputs and errors. Compute the mean square error over all inputs according to Eq. (9).
- Compute the Jacobian matrix, J(x), where x represents the weights and biases of the network.
- Then the weights are updated using the formula

$$w_{t+1} = w_t - \left(J^T(x)J(x) + \mu I\right)^{-1} J^T(x)E \quad (9)$$

where, $J^T(x)J(x)$ is referred to as the Hessian matrix, μ is a training parameter, I is the identity unit matrix and E is a vector of size kl calculated as:

$$E = \left[t_{11} - y_{11}t_{12} - t_{12}...t_{kl} - t_{kl}\right]^T \quad (10)$$

- Recompute the error. If this new error is smaller than that computed in step 1, then decrement the training parameter μ, recompute the error and go back to step 1. If the error is not reduced, then increase the training parameter μ and go back to step 3. The decrement and increment value of μ are predefined by the user. Normally the increment value of μ is set to 10 and the decrement value of μ is set to 0.1.
- The algorithm is converged when the gradient is less than some predetermined value, or when the mean square error has been reduced to predetermined value or maximum number of epochs has been reached. The maximum number of epochs is predefined by the user.

For $\mu = 0$ the algorithm becomes Gauss-Newton method. For very large μ the LM algorithm becomes steepest decent or the error backpropagation algorithm. The parameter is automatically adjusted at each iteration in order to secure convergence. The LM algorithm requires computation of the Jacobian J(x) matrix at each iteration step and the inversion of $J^T(x)J(x)$ square matrix.

STRUCTURE ANALYSIS OF MLP NETWORK

The recognition performance of the MLP network will highly depend on the structure of the network and training algorithm. In the proposed system, the multilayer perceptron network was trained using various algorithms. The algorithm which has given much better convergence speed and good recognition rate has been selected to train the network.

The number of nodes in input, hidden and output layers will determine the network structure. The hidden and output nodes have activation function that will also influence the network performance. The best network structure is normally problem dependent, hence structure analysis has to be carried out to identify the optimum structure. In the proposed system, the number of input and output nodes were fixed at 80 and 7, respectively, since the feature extracted from the handwritten character images were the 8 invariant fourier descriptors, eight transition features, 16 average pixel values, 16 tenure feature values, 16 angles and 16 distance values and the target outputs are Tamil characters denoted in binary form. Therefore, the number of hidden nodes and activation functions are to be determined. The time taken to train the network, recognition efficiency and Mean square error will be used to judge the performance of the MLP network.

Training algorithm: Multilayer perceptron network trained using classical Backpropagation network algorithm (CBPN), Resilent Backpropagation Neural Network algorithm (RBPN), Conjugate Gradient Backpropagation Neural Network algorithm (CGBPN), Scaled Conjugate Gradient Backpropagation Neural Network algorithm (SCGBPN), Quasi- Newton Backpropagation Neural Network algorithm (QNBPN) and Levenberg-Marquardt (LM) algorithm are used for recognition. The results of the experimentation are shown in Table 2. The recognition accuracy for train set varies from 87.9-98.9%. The recognition accuracy for test set varies from 77.4-94.1%. It shows that LM algorithm converges faster than other algorithms and also gives high recognition rate compared to other algorithms. Then multilayer perceptron network with one hidden layer trained using Levenberg-Marquardt algorithm is selected for recognition.

Table 2: Results for various training algorithms

No. of hidden nodes	MSE	Training time (in secs)	Testing time (in secs)	Recognition rate	
				Train set	Test set
5	0.2151	118	0.0150	71.1	65.5
10	0.1733	247	0.0150	78.2	70.2
15	0.1466	388	0.0310	83.5	73.1
20	0.1299	446	0.0160	85.1	78.3
25	0.0988	524	0.0150	87.1	82.2
30	0.0711	655	0.0160	91.3	85.3
35	0.0574	846	0.0150	94.2	87.5
40	0.0483	1012	0.0150	95.6	89.1
45	0.0401	1266	0.0160	96.2	92.3
50	0.0314	1512	0.0150	98.9	94.1

Table 3: MSE, Recognition rate variation with no. of hidden nodes

Algorithm	Train time (in secs)	Test time (in secs)	MSE	Recognition rate	
				Train set (%)	Test set (%)
CBPN	3437	0.0160	0.1638	89.5	77.4
RBPN	3709	0.0310	0.1497	88.4	79.3
CGBPN-CGF	2229	0.0320	0.1470	87.9	78.6
CGBPN-CGP	7524	0.0160	0.1441	89.1	79.6
CGBPN-CGB	1984	0.0320	0.1364	90.6	81.3
SCGBPN	6610	0.0310	0.1397	89.5	80.7
QNBPN-BFG	2553	0.0460	0.1385	90.5	81.1
QNBPN-OSS	6055	0.0160	0.1342	90.9	82.4
LM	1512	0.0150	0.0314	98.9	94.1

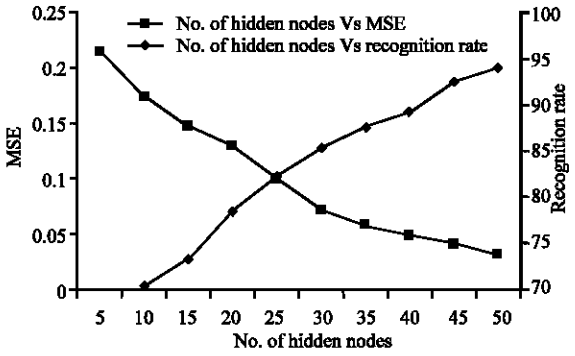

Fig. 7: MSE, recognition rate variation with no. of hidden nodes

Number of hidden layer nodes: The number of hidden nodes will heavily influence the network performance. In this analysis, the number of hidden nodes was increased from 5 to 50, the time taken to train the network in seconds, recognition efficiency for training data and test data and mean square error as in Table 3 were obtained. It was obvious from the table that if the number of hidden nodes increases the recognition efficiency is also increases.

Increasing number of hidden nodes beyond 50 takes more training time but could not more beneficial in the recognition efficiency. MSE and Recognition efficiency variation with different number of Hidden nodes is plotted in the Fig. 7.

Table 4: MSE, Recognition rate variation with different activation function of hidden nodes

Activation function of output nodes	MSE	Training time (in sec)	Testing time (in sec)	Recognition rate	
				Train set	Test set
Logsing	0.0343	931	0.0160	95.2	90.2
Tansig	0.0342	931	0.0160	95.6	92.7
Purelin	0.0314	1512	0.0150	98.9	94.1
Satlin	0.0390	678	0.0150	94.1	87.2
Hardlim	0.2412	118	0.0160	87.2	75.8

Type of activation function: The hidden nodes and output nodes have activation function it will also influence the network performance. In this analysis, the number of hidden nodes was taken as 50 and the maximum epoch was set to 150. The activation functions for the output nodes were selected as pure linear function and the activation function of the hidden nodes were changed, the time taken to train the network in seconds, recognition efficiency for training data and test data and mean square error as in Table 4 were obtained.

The results showed that the most suitable activation function for hidden nodes was tangent sigmoid function and hence tangent sigmoid function has been selected as the activation function of the hidden nodes.

A similar analysis was carried out to determine the most suitable activation function for output nodes. In this case, the activation functions for the hidden nodes were selected as tangent sigmoid and the activation function of the output nodes were changed, the time taken to train the network in seconds, recognition efficiency for training data and test data and mean square error as in Table 5 were obtained.

The results showed that the most suitable activation function for output nodes was pure linear function, hence pure linear function has been selected as the activation function of the output nodes.

Number of training epochs: Number of training epochs taken to train a network also plays an important role in determining the performance of neural network. The objective is to recognise the handwritten Tamil characters, then the number of training epochs after which the network is capable to recognise more than 90% characters accurately can be considered as sufficient. In this proposed system training epoch is set to 150, which could be sufficient to recognize 94.1% characters accurately. Increasing the number of training epochs took more training time but could not beneficial in the recognition efficiency.

Table 5: MSE, Recognition rate variation with different activation function of output nodes

Activation function of output nodes	MSE	Training time (in secs)	Testing time (in secs)	Recognition rate	
				Train set	Test set
Logsig	0.0319	1492	0.0160	97.2	90.1
Tansig	0.0314	1512	0.0150	98.9	94.1
Purelin	0.2358	163	0.0310	80.3	75.2
Satlin	0.0761	540	0.0160	90	85.2
Hardlim	0.2499	244	0.0160	79.1	71.7

EXPERIMENTAL RESULTS

The invariant Fourier descriptors feature is independent of position, size and orientation. Finding of transition values for each character is to avoid misclassification due to rotational invariance and average pixel values of each character could be very much useful to distinguish the characters with similar shape. With the combination of Fourier descriptors, Transition features, average pixel values, tenure values, distance and angle features and LM algorithm, a high accuracy recognition system is realized. The training set contained 3150 characters with 30 samples from each character class. All the trained characters were written by 30 different writers on a pre-formatted paper. The test set contained a separate set of 1575 characters. A portion of the training data was also used to test the system. In the training set, more than 95% of recognition rate was achieved and the recognition rate for test set varies between 90-94.1%.

An analysis was carried out to determine the suitable training algorithm, number of hidden layer neurons, activation function for hidden as well as output nodes and number of training epochs to achieve high performance of MLP network in the recognition of handwritten Tamil characters. It suggested that the higher recognition rate was achieved for the MLP network trained using LM algorithm with 50 hidden nodes with tangent sigmoid function for hidden nodes, pure linear function for output nodes with a maximum training epoch of 150.

CONCLUSION

In this study, we have presented a system for recognizing handwritten Tamil characters. The methods described here for processing of Tamil handwritten characters can also be used for other Indian scripts and numerals. Experimental results shows that Fourier descriptors, Transition features, average pixel values, tenure values, distance and angle features combined with MLP network trained using LM algorithm yields good recognition accuracy. This work can be further extended by including few other preprocessing activities like slant correction and segmentation of touching and kerned characters. This research is still ongoing and many improvements and additions to preprocessing and feature extraction and recognition techniques shall be explored.

REFERENCES

Aparna, K.H., V. Subramanian, M. Kasirajan, G. Vijay Prakash, V.S. Chakravarthy and S. Madhvanath, 2004. Online Handwriting Recognition for Tamil. Proceedings of the 9th IEEE International Workshop on Frontiers in Handwriting Recognition, IWFHR-9.

Azimi-Sadjadi, M.R., S. Citrin and S. Sheedvash, 1990. Supervised learning process of multi-layer perceptron neuralnetworks using fast least squares. Proc. Int. Conf. Acoustics, Speech and Signal Processing. ICASSP, 3: 1381-1384.

Bansal, V. and R.M.K. Sinha, 1999. On how to describe shapes of Devanagari Characters and use them for Recognition. Proc. 5th Int. Conf. Document Analysis and Recognition, Bangalore, India, pp: 410-413.

Blumenstein, M. and B. Verma, 1997. A segmentation algorithm used in conjunction with Artificial neural networks for the recognition of real-world Postal addresses. Proc. Int. Conf. Computational Intell. Multimedia Applic., ICCIMA, pp: 155-160.

Blumenstein, M. and B. Verma, 1998. A neural Based Segmentation and Recognition Technique for Handwritten Words. Proc. IEEE Int. Joint Conf. Neural Networks, 3: 1738-1742.

Bozinovic, R.M. and S.N. Srihari, 1989. Off-line cursive script word recognition. IEEE. Trans. Pattern Anal. Machine Intell., 11(1): 68-83.

Brijesh Verma, 1997. Fast training of Multilayer Perceptrons. IEEE. Trans. Neural Networks, 8 (6): 1314-1319.

Chinnuswamy, P. and S.G. Krishnamoorthy, 1980. Recognition of handprinted Tamil characters. Pattern Recognition, 12: 141-152.

Chung, Y.Y. and M.T. Wong, 1997. Handwritten Character Recognition by Fourier Descriptors and Neural Network Proc. IEEE TENCON- Speech and Image Technologies for Computing and Telecommunications.

Deng, D., K.P. Chan and Y. Yu, 1994. Handwritten Chinese Character Recognition using Spatial Gabor filters and Self-organizing Feature Maps Proc. IEEE Int. Conf. Image Processing, 3: 940-944.

Ha, T.M. and H. Bunke, 1997. Offline Handwritten Numeral Recognition by Perturbation Method. IEEE Trans. Pattern Anal. Machine Intell., 19(5): 535-539.

Hanmandlu, M. and O.V. Ramana Murthy, 2005. Fuzzy Model Based Recognition of Handwritten Hindi Numerals. Proc. Int. Conf. Cognition Recog., pp: 490-496.

Hanmandlu, M., K.R.M. Mohan and H. Kumar, 1999. Neural-based handwritten character recognition. Proceedings of 5th IEEE International Conference on Document Analysis and Recogniton, ICDAR.

Hewavitharana, S. and H.C. Fernando, 2002. A two Stage Classification Approach to Tamil Handwriting Recognition. Proc. the Tamil Internet Conf. California, U.S.A., pp: 118-124.

Hongbong, K. and N. Kwanghee, 1995. Object Recognition of One-DOF Tools by a Back-Propagation Neural Net. IEEE. Trans. Neural Networks, 6 (2): 484-487.

Lee, S.W., 1996. Off-line Recognition of Totally Unconstrained Handwritten Numerals using Multilayer Cluster Neural Network. IEEE Trans. Pattern Anal. Machine Intell., 18 (6): 648-652.

Michael, T.Y.L. and Y.S. Ching, 1981. Automatic Recognition of characters by fourier descriptors and boundary line encodings. Pattern Recognition, 14: 383-393.

Niranjan, J., G. Sita, A.G. Ramakrishnan and Sriganesh Madhvanath, 2004. Comparison of Elastic Matching Algorithms for Online Tamil Handwritten Character Recognition. Proceedings of the 9th IEEE International Workshop on Frontiers in Handwriting Recognition, IWFHR-9.

Park, D.J., B.E. Jun and J.H. Kim, 1992. Novel Fast Training Algorithm for Multilayer Feedforward neural network. Elec. Lett., 28(6): 543-544.

Shanthi, N. and K. Duraiswamy, 2005. Preprocessing Algorithms for the Recognition of Tamil Handwritten Characters. 3rd Int. CALIBER, Cochin, pp: 77-82.

Suresh, R.M., S. Arumugam and L. Ganesan, 1999. Fuzzy Approach to Recognize Handwritten Tamil characters, Published by IEEE, ICCIMA, New Delhi, pp: 459-464.

Vasantha Lakshmi, C. and C. Patvardhan, 2003. A high accuracy OCR system for Printed Telugu Text. Proceed. Conf. Convergent Technologies for Asia-Pacific Region, TENCON, 2: 725-729.