Hand Written Character Feature Extraction using Non-Linear Feedforward Neural Networks

A. JALIL, I. M. QURESHI, T. A. CHEEMA* AND A. NAVEED*
Engineering and Computer Sciences
M. A. Jinnah University
Islamabad, Pakistan
E-mail: a_jalil@yahoo.com
*Department of Electronics
Quaid-i-Azam University
Islamabad, Pakistan
E-mail: imq313@yahoo.com

In this paper, an artificial neural network is proposed for feature extraction of hand written characters. The learning algorithm is developed based on a proposed modified Sammon’s stress for our feedforward neural networks, which can not only minimize intra class pattern distances but also preserve interclass distances in the output feature space. The proposed feature extraction method tries to calculate rough classes using a Competitive Learning neural network, which is an unsupervised neural network. Then the proposed neural network was used with modified Sammon’s stress to perform feature extraction using information obtained by means of a Competitive Learning Network. The features thus obtained were compared with a standard PCA neural network and a neural network using Sammon’s stress in terms of their classification accuracy. Two numerical criteria were used for performance evaluation of the features – the normalized classification error rate and modified Sammon’s stress. It is found that proposed modified Sammon’s stress provides features that are more efficient based on these two numerical criteria.

Keywords: feature extraction, unsupervised neural networks, Sammon’s stress, hand-written character recognition, LMS algorithm

1. INTRODUCTION

Handwritten character recognition is one of the important fields of pattern recognition. The problem becomes more difficult in case of severely degraded, omni-font machine printed or hand written characters [1]. Character recognition involves the performance of two difficult tasks. One of them is feature extraction, and the other is classification. Feature extraction is an information reduction process and involving extraction of the most important information from raw data for classification purpose. The information thus obtained must satisfy the requirement that intra-class pattern variability is mini-
mized while inter-class pattern variability is enhanced [2, 3]. Feature extraction can avoid the “curse of dimensionality” [4] improving the generalization ability of classifiers and reducing the computational requirements of pattern classification [5]. Therefore, different classical feature extraction methods such as Geometrical moments, Zernike moments and Fourier descriptors and adaptive methods such as neural networks, have been proposed in the literature. They fulfill the above requirement to varying degrees depending on the specific recognition problem and available data [6-9]. We have proposed a feature extraction method that uses neural networks to reduce the computational complexity and speed of classifiers while also solving the problems encountered with degraded and omni font handwritten characters.

A large number of approaches to feature extraction are available. They use neural networks that differ from each other in terms of their mapping function from high to lower dimension as well as their learning and optimization criteria [10, 11]. The mapping technique may be linear or nonlinear, and the network may be supervised or unsupervised. Therefore, four types of neural networks exist. These are supervised linear networks, such as the Linear Discriminate Analysis neural network, supervised nonlinear networks, such as the Nonlinear Discriminate Analysis neural network, unsupervised linear networks, such as the Principal Component Analysis neural network and unsupervised nonlinear networks, such as Sammon artificial neural network. These linear methods are attractive because they require fewer computations than nonlinear methods, but the latter are more powerful than linear methods. Although supervised methods provide better features, these features are not usually available in real situations [10]. Therefore, unsupervised nonlinear methods are considered to be the most powerful tools for feature extraction.

Sammon [12] proposed a non-linear mapping technique for data structure analysis, which attempts to maximally preserve inter-pattern distances. Let \( x(\mu), \mu = 1, 2, \ldots, n \) be the \( n \) \( N \)-dimensional patterns in the input space, and let \( y(\mu), \mu = 1, 2, \ldots, n \) be the \( n \) corresponding patterns in the \( M \)-dimensional projected space. Feature extraction is a mapping \( \phi \) from \( N \)-dimension input space to \( M \)-dimension output space (projected space):

\[ \phi: R^N \to R^M, M < N \]  

such that some criterion, \( C \), is optimized. Sammon proposed a criterion known as Sammon’s stress, which is to be minimized:

\[ E = \frac{1}{2} \sum_{\mu=1}^{n-1} \sum_{\nu=\mu+1}^{n} \frac{S^*(\mu, \nu) - S(\mu, \nu)}{S^*(\mu, \nu)} \sum_{\mu=1}^{n-1} \sum_{\nu=\mu+1}^{n} S^*(\mu, \nu) \]  

where \( S^*(\mu, \nu) \) is the distance between pattern \( \mu \) and pattern \( \nu \) in the input space and can be expressed as

\[ S^*(\mu, \nu) = \left[ \sum_{n=1}^{N} (x_n(\mu) - x_n(\nu))^2 \right]^{1/2}. \]
$N$ is the dimension of the pattern $\mu$ and pattern $\nu$ in the input space, and $S(\mu, \nu)$ is the distance between pattern $\mu$ and pattern $\nu$ in the projected space and can be expressed as

$$S(\mu, \nu) = \left[ \sum_{m=1}^{M} (y_m(\mu) - y_m(\nu))^2 \right]^{1/2},$$

where $M$ is the dimension of the pattern $\mu$ and $\nu$ in the projected space.

The Sammon stress $E$ is the measure of how well the inter-pattern distances are preserved when the patterns are projected from a high dimensional space to a lower dimensional space. Sammon used a gradient descent algorithm to find a configuration that attempts to minimize the Sammon’s stress.

Mao and Jain [13] proposed a non-linear projection algorithm that uses a neural network for feature extraction and data projection using Sammon’s stress. The technique used to find a feature vector and data projection attempts to maximally preserve inter-pattern distances. It has the generalization ability but at the same time has serious drawbacks. There are many local minima in the energy surface, and the algorithm will inevitably get stuck in a local minima. One usually runs the algorithm several times with different random initial configurations and chooses the configuration with the lowest stress. Another disadvantage is the large amount of computation required [14, 15].

Mao and Jain [5] redefined their algorithm by implementing Sammon’s projection technique in an unsupervised feedforward neural network with a learning algorithm based on the steepest descent criterion with Sammon’s stress taken as the error surface [16]. Their proposed neural network produces much better data projection and analysis results. They inspected and compared the results produced by the network with those produced by other networks such as the Linear Discriminate Analysis neural network [17, 18], Nonlinear Discriminate Analysis neural network [2] and Principal Component Analysis neural network [19, 20]. They visualized the results by transforming different higher dimensional data sets into two dimensional outputs. They concluded that Sammon’s mapping technique produces much better results for exploratory data analysis but not feature extraction because it preserves inter pattern distances irrespective of whether patterns are inter-class or intra-class.

In this paper, we propose a feedforward neural network model for feature extraction of handwritten characters based on a modified Sammon’s stress which not only preserves inter-class pattern distances but also minimizes intra-class pattern distances. This is the modified Sammon’s stress, which is the main contribution of this paper. The gradient-based algorithm is proposed for the upgrading of weights of neural networks [21]. The results produced by the proposed neural network are also compared with those produced by the Principal Components Analysis Neural Network and unsupervised feedforward neural network using the standard Sammon’s stress. Section 2 presents a proposed feature extraction method in two parts. Part one, explains the behavior of the unsupervised competitive learning neural network (CLN), used to perform rough classification of data [22, 23]. This information is provided to the feedforward neural network with the proposed Sammon’s stress, which is explained in the part two. Principal Component Analysis neural network is used for comparison, is elaborated in section 3. Sec-
tion 4 contains comparison of the results of proposed neural network, Principal Components Analysis Neural Network and unsupervised feedforward neural network using the standard Sammon’s stress. Section 5 presents conclusions.

2. FEATURE EXTRACTION USING NEURAL NETWORKS

The features that are most important for classification purposes should be preserved, and the information that does not fulfill this criterion should be treated as redundant information and removed. The features of objects should also reduce the computational complexity of the classifiers. We have proposed an architecture for neural networks based on the above criteria for the extraction of features of handwritten characters. Therefore, a set of handwritten characters, used for training, was presented to an unsupervised neural network called the competitive learning neural network (CLN) for classification of these characters. This classification information was then used by our feedforward neural network. This network uses our proposed modified Sammon’s stress for learning purposes in order to get more refined features, and this is the main contribution of the paper. Once the proposed modified Sammon’s Stress is minimized, the feedforward neural network is trained. Then it can be used to perform feature extraction on handwritten characters.

2.1 Competitive Learning Network

This competitive learning network generates classes of given data. This neural network is unsupervised, having an input layer and a competition or output layer as shown in Fig. 1. The input layer neurons are fully connected to the output layer with feedforward connections while output neurons have self-excitatory and inhibitory connections as well. Competitive networks have two phases [24, 25]. In the first phase a winner is selected, while in the second phase, the winner is rewarded by updating the weights of the input to the output layer. The output of the $j^{th}$ neuron, $O_j$, for the $i^{th}$ input vector $x_i$ using feedforward connections $\xi_{ij}$ is

$$O_j = \sum \xi_{ij} x_i.$$  

Picking out a single winner following the evaluation of the inner product $O_j$ over all values of $j$ can be done by using lateral connection weights in the connection layer, which has $M$ output units. The lateral connections are as follows:

$$\zeta_{hk} = \begin{cases} 1, & h = k \\ -\delta, & h \neq k, \text{ where } \delta < \frac{1}{M} \text{ and } 1 \leq h, k \leq M. \end{cases}$$  

$y_j(k)$ is the output of unit $j$ at the time index $k$. $y_j(0)$ is found after the input vector is removed, and it no longer affects the state transitions in the output layer. Each unit in the output layer has a sigmoid activation function with zero thresholds. It evolves according to the following equation:
HANDWRITTEN CHARACTER FEATURE EXTRACTION

\[ y_j(k+1) = T_f \left( y_j(k) - \delta \sum_{r \neq j} y_r(k) \right), \quad y_j(0) = T_f(0_j), \]  

(7)

where \( T_f \) is a sigmoid function and \( 0 \leq T_f(x) \leq 1 \) for all real values of the argument \( x \). Once a winner has been picked, the weight vector \( \xi_j \) of the winner is updated so as to bring it even closer to \( x_i \) so that the winning unit is more likely to win next time if the same or similar input vectors are presented. The update has the effect of moving the connection vector \( \xi_j \) towards the center of the cluster to which it belongs. The weight vector for the winner is updated using the following equation:

\[ \xi_j(k+1) = \beta (\xi_j(k) + \alpha(k)(x_i - \xi_j(k))), \]  

(8)

where \( 0 \leq \alpha(k) \leq 1 \) is the learning rate at the time index \( k \) and \( \beta \) is the normalization factor, given by

\[ \beta = \left\| \xi_j(k) + \alpha(k)(x_i - \xi_j(k)) \right\|^{-1}. \]  

(9)

This neural network is used to conduct rough classification of the input data. This rough classification information is provided to the next feedforward neural network, which utilizes the modified Sammon’s stress to find feature vectors with better performance as compared to other schemes.

2.2 Proposed Modified Sammon’s Stress and Learning Algorithm

The proposed neural network for feature extraction is a feedforward neural network in which neurons are arranged in the form of layers. Generally, all neurons in layers are connected to all neurons in adjacent layers through uni-directional links. These links are represented by synaptic weights [26]. The proposed architecture consists of one input
layer, two hidden layers and one output layer as shown in Fig. 2. The \( j \)th output of the neuron, \( u_j \), of any layer is the weighted sum of all the inputs:

\[
\begin{align*}
\text{l}1_n, \text{s} = & \sum_{i=0}^{n_{-1}} w_{ij}^{[s]} \cdot o_{i}^{[s-1]} \\
\text{where } w_{ij}^{[s]} & \text{ represents the synaptic weights of the } j \text{th neuron which are multiplied by the inputs } \text{o}_{i}^{[s-1]} \text{ to obtain outputs } o_{i}^{[s]}, \text{n}_{s} \text{ is the number of neurons in the } s \text{th layer and } s = 1, 2, 3. \text{ The unipolar sigmoid function } \psi(u) \text{ whose range is } (0.0, 1.0), \text{ is used for each unit. The output of the } j \text{th unit in the } s \text{th layer can be written as}
\end{align*}
\]

\[
\begin{align*}
y_{j}^{[s]} = & \psi \left( \sum_{l=0}^{n_{-1}} w_{jl}^{[s]} \cdot y_{l}^{[s-1]} \right),
\text{where } y_{j}^{[s-1]} = o_{l}^{[s-1]}.
\end{align*}
\]

We propose an error function \( E \), which we call the modified Sammon’s stress. It is used to update the values of synaptic weights. By means of class information from the competitive learning neural network (CLN), the interclass pattern distances are enhanced or preserved while the inter-pattern distances of the same class are minimized. The modified Sammon’s stress is

\[
E = \sum_{k=1}^{m-1} \sum_{l=k+1}^{m} E_{kl}(1 - \delta_{kl}) + \sum_{k=1}^{m} E_{kk} \left\{ \begin{array}{ll}
\delta_{kl} = 1 & \text{for } k = l \\
\delta_{kl} = 0 & \text{for } k \neq l,
\end{array} \right.
\]

Fig. 2. The Proposed feed forward neural network.
where $m$ is the total number of classes and $E_{kl}$ is the interclass error between class $k$ patterns and class $l$ patterns. Class $k$ has a total of $k_1$ patterns, and class $l$ has total of $l_1$ patterns, and is defined as

$$E_{kl} = \sum_{\mu=1}^{k_1} \sum_{\nu=1}^{l_1} E_{k,\ell}(\mu, \nu),$$  \hspace{1cm} (13)$$

where

$$E_{k,\ell}(\mu, \nu) = \lambda \frac{S_{kl}(\mu, \nu) - \rho S_{kl}(\mu, \nu)}{S_{kl}(\mu, \nu)} - \lambda,$$  \hspace{1cm} (14)$$

and

$$\lambda = \frac{1}{\sum_{\mu=1}^{k_1} \sum_{\nu=1}^{l_1} S_{kl}(\mu, \nu)}$$  \hspace{1cm} and  \hspace{1cm} 0 < \rho \leq 1. \hspace{1cm} (15)$$

The error $E_{k,\ell}(\mu, \nu)$ is proportional to the change in inter-pattern distance due to the projection from the $N$-dimensional original space to the $M$-dimensional projected space when patterns $\mu$ and $\nu$ belong to different classes. $S_{kl}(\mu, \nu)$ is the distance between patterns $\mu$ and $\nu$ in the original space. The patterns $\mu$ and $\nu$ belong to different classes $k$ and $l$, respectively, and can be expressed as

$$S_{kl}(\mu, \nu) = \left[ \sum_{n=1}^{N} (x_{k,n}(\mu) - x_{l,n}(\nu))^2 \right]^{1/2}. \hspace{1cm} (16)$$

$S_{kl}(\mu, \nu)$ is distance between patterns $\mu$ and $\nu$ in the projected space. These patterns belong to different classes $k$ and $l$, respectively, and can be expressed as

$$S_{kl}(\mu, \nu) = \left[ \sum_{n=1}^{M} (y_{k,n}(\mu) - y_{l,n}(\nu))^2 \right]^{1/2}. \hspace{1cm} (17)$$

In Eq. (12), $E_k$ is the intra-class error of class $k$ with a total of $k_1$ patterns and can be defined as

$$E_k = \sum_{\mu=1}^{k_1} \sum_{\nu=\mu+1}^{k_1} E_k(\mu, \nu),$$  \hspace{1cm} (18)$$

where

$$E_k(\mu, \nu) = \lambda \frac{(S_k(\mu, \nu))^2}{S_k(\mu, \nu)} - \lambda,$$  \hspace{1cm} (19)$$

and

$$\lambda = \frac{1}{\sum_{\mu=1}^{k_1} \sum_{\nu=\mu+1}^{k_1} S_k(\mu, \nu)}. \hspace{1cm} (20)$$
\( E_k(\mu, \nu) \) is proportional to the inter-pattern distance in the \( M \)-dimension projected space where patterns \( \mu \) and \( \nu \) belong to the same class. \( S_k^*(\mu, \nu) \) is the distance between patterns \( \mu \) and \( \nu \) in the original space belonging to class \( k \) and can be expressed in Eq. (21),

\[
S_k^*(\mu, \nu) = \left[ \sum_{i=1}^{N} (x_{k,i}(\mu) - x_{k,i}(\nu))^2 \right]^{1/2}.
\] (21)

\( S_k(\mu, \nu) \) is the distance between patterns \( \mu \) and \( \nu \) in the projected space belonging to class \( k \) and can be expressed as

\[
S_k(\mu, \nu) = \left[ \sum_{i=1}^{M} (y_{k,i}(\mu) - y_{k,i}(\nu))^2 \right]^{1/2}.
\] (22)

Note that our proposed modified Sammon’s stress consists of two error functions, which are not active simultaneously. Once we have two patterns, competitive neural network (CLN) provides us with class information. If these two patterns belong to same class, the error function \( E_k \) is activated; otherwise, the error function \( E_{k,l} \) is activated. Although the class information is used to select one of the two error functions, there is no known desired response at the output as in the case of a supervised neural network with a backpropagation algorithm. Hence, it may not be a strictly supervised neural network.

A steepest descent method is proposed here to update the synaptic weights of the neural network using the proposed modified Sammon’s stress given in Eq. (12). The synaptic weights for the third layer for patterns \( \mu \) and \( \nu \) of different classes \( k \) and \( l \) are given as

\[
w_{ij}^{[3] \text{new}} = w_{ij}^{[3] \text{old}} - \eta \frac{\partial E_{k,l}(\mu, \nu)}{\partial w_{ij}^{[3]}}.
\] (23)

By applying the chain rule, the second part of Eq. (23) as follows

\[
\frac{\partial E_{k,l}(\mu, \nu)}{\partial w_{ij}^{[3]}} = \sum_{n=1}^{M} \left[ \frac{\partial E_{k,l}(\mu, \nu)}{\partial S_{k,l}(\mu, \nu)} \frac{\partial S_{k,l}(\mu, \nu)}{\partial (y_{k,n}(\mu) - y_{k,n}(\nu))} \frac{\partial (y_{k,n}(\mu) - y_{k,n}(\nu))}{\partial w_{ij}^{[3]}} \right]
\] (24)

where \( M \) is the number of units in the third layer.

\[
\frac{\partial E_{k,l}(\mu, \nu)}{\partial w_{ij}^{[3]}} = -2\lambda \left[ S_{k,l}(\mu, \nu) - S_{k,l}(\mu, \nu) \right] \left[ y_{i,j}^{[2]}(\mu) - y_{i,j}^{[2]}(\nu) \right] \times \left[ \psi'(u_{j,\mu,k}) y_{i,j}^{[2]}(\mu) - \psi'(u_{j,\nu,k}) y_{i,j}^{[2]}(\nu) \right].
\] (25)
where \( u_{j,\mu,k} \) is the output of the \( j \)th unit in the third layer of pattern \( \mu \) of class \( k \), and \( u_{j,\nu,l} \) is the output of \( j \)th unit in the third layer of pattern \( \nu \) of class \( l \), and

\[
\psi'(u_{j,q,p}) = \gamma(1 - y_{p,j}^{[3]}(q))y_{p,j}^{[3]}(q), \quad p = k \text{ or } l \text{ and } q = \mu \text{ or } \nu,
\]

which is the derivative of unipolar sigmoid function, which is given as

\[
\psi(u_{j,q,k}) = \frac{1}{1+\exp(-\gamma u_{j,q,p})},
\]

where \( \gamma \) is the positive constant or variable which controls the slope of the sigmoid function. Let

\[
d_{kj}^{[3]}(\mu, \nu) = -2\rho\gamma\frac{\left[ S_{kl}^{[3]}(\mu, \nu) - S_{kl}^{[3]}(\mu, \nu) \right] y_{k,j}^{[3]}(\mu) - y_{l,j}^{[3]}(\nu)}{S_{kl}^{[3]}(\mu, \nu)S_{kl}^{[3]}(\mu, \nu)}
\]

\[
\delta_{p,j}^{[3]}(q) = d_{kj}^{[3]}(\mu, \nu)(1 - y_{p,j}^{[3]}(q))y_{p,j}^{[3]}(q), \quad p = k \text{ or } l \text{ and } q = \mu \text{ or } \nu
\]

where \( d_{kj}^{[3]}(\mu, \nu) \) is the change in the output scaled by the normalized interpattern distance change when we present pattern \( \mu \) belonging to class \( k \) and pattern \( \nu \) belonging to class \( l \). Consequently \( \delta_{p,j}^{[3]}(\mu) \) and \( \delta_{p,j}^{[3]}(\nu) \) will be backpropagated to layer 2. Substituting Eq. (26), Eq. (28) and Eq. (29) into Eq. (25), we get

\[
\frac{\partial E_{k,j}(\mu, \nu)}{\partial w_{qj}^{[3]}} = \delta_{k,j}^{[3]}(\mu)y_{k,j}^{[2]}(\mu) - \delta_{l,j}^{[3]}(\nu)y_{l,j}^{[2]}(\nu).
\]

The updating rule for the third layer is obtained by substituting Eq. (30) into Eq. (23), we get

\[
w_{qj}^{[3]_{new}} = w_{qj}^{[3]_{old}} - \eta \left[ \delta_{k,j}^{[3]}(\mu)y_{k,j}^{[2]}(\mu) - \delta_{l,j}^{[3]}(\nu)y_{l,j}^{[2]}(\nu) \right].
\]

The weights of the second layer of the neural network were updated using the following equation:

\[
w_{qj}^{[2]_{new}} = w_{qj}^{[2]_{old}} - \eta \left[ \delta_{k,j}^{[2]}(\mu)y_{k,j}^{[1]}(\mu) - \delta_{l,j}^{[2]}(\nu)y_{l,j}^{[1]}(\nu) \right],
\]

where

\[
\delta_{p,j}^{[2]}(q) = d_{p,j}^{[2]}(q)(1 - y_{p,j}^{[2]}(q))y_{p,j}^{[2]}(q), \quad p = k \text{ or } l \text{ and } q = \mu \text{ or } \nu,
\]

and

\[
d_{p,j}^{[2]}(q) = \sum_{n=1}^{m} \delta_{p,n}(q)w_{jm}^{[3]}, \quad p = k \text{ or } l \text{ and } q = \mu \text{ or } \nu,
\]
where \( m \) is the number of units in the 3\(^{\text{rd}}\) layer,

\[
\delta^{(3)}_{p,n}(q) = \gamma^2 d^{(3)}_{p,n} (\mu, \nu) (1 - y^{(3)}_{p,n}(q)) y^{(3)}_{p,n}(q), \quad p = k \text{ or } l \text{ and } q = \mu \text{ or } \nu.
\]

(35)

The weights of the first layer of the neural network are updated using the following equation:

\[
w^{[1]}_{ij} = \frac{\partial w_{ij}^{(t)}}{\partial \tilde{w}_{ij}} - \eta \left[ \delta^{(3)}_{i,j}(\mu) y^{(0)}_{i,j}(\mu) - \delta^{(3)}_{l,j}(\nu) y^{(0)}_{l,j}(\nu) \right],
\]

(36)

where

\[
\delta^{(1)}_{p,j}(q) = d^{(1)}_{p,j}(q)(1 - y^{(1)}_{p,j}(q)) y^{(1)}_{p,j}(q), \quad p = k \text{ or } l \text{ and } q = \mu \text{ or } \nu,
\]

(37)

\[
d^{(1)}_{p,j}(q) = \sum_{n=1}^{m} s_{p,n}^{(2)}(q) w_{jm}^{(2)}, \quad p = k \text{ or } l \text{ and } q = \mu \text{ or } \nu,
\]

(38)

where \( m \) is the number of units in the 2\(^{\text{nd}}\) layer and

\[
\delta^{(2)}_{p,n}(q) = \gamma^2 d^{(2)}_{p,n} (\mu) (1 - y^{(2)}_{p,n}(q)) y^{(2)}_{p,n}(q), \quad p = k \text{ or } l \text{ and } q = \mu \text{ or } \nu.
\]

(39)

The synaptic weights of third layer for pattern \( \mu \) and \( \nu \) of the same class \( k \) are given as

\[
w^{[3]}_{ij} = \frac{\partial E_k(\mu, \nu)}{\partial w_{ij}^{[3]}} - \eta \frac{\partial E_k(\mu, \nu)}{\partial w_{ij}^{[3]}},
\]

(40)

\( E_k(\mu, \nu) \) is the mapping error defined by Eq. (19). It is proportional to the interpattern distance between patterns \( \mu \) and \( \nu \) of the same class \( k \), due to the projection from the \( N \)-dimension original space to the \( M \)-dimension projected space:

\[
\frac{\partial E_k(\mu, \nu)}{\partial w_{ij}^{[3]}} = \sum_{n=1}^{M} \left[ \frac{\partial E_k(\mu, \nu)}{\partial S_k(\mu, \nu)} \frac{\partial S_k(\mu, \nu)}{\partial (y^{[3]}_{k,n}(\mu) - y^{[3]}_{k,n}(\nu))} \frac{\partial (y^{[3]}_{k,n}(\mu) - y^{[3]}_{k,n}(\nu))}{\partial w_{ij}^{[3]}} \right]
\]

(41)

\[
= -2\lambda \left[ y^{[3]}_{k,j}(\mu) - y^{[3]}_{k,j}(\nu) \right] \left[ \psi'(u_{j,\mu,k}) y^{[2]}_{k,j}(\mu) - \psi'(u_{j,\nu,k}) y^{[2]}_{k,j}(\nu) \right]
\]

(42)

where \( u_{j,\mu,k} \) is the output of \( j^{\text{th}} \) unit in the third layer of pattern \( \mu \) of class \( k \) and \( u_{j,\nu,l} \) is the output of \( j^{\text{th}} \) unit in the third layer of pattern \( \nu \) of same class \( k \) and

\[
\psi'(u_{j,\mu,k}) = \gamma (1 - y^{[3]}_{k,j}(q)) y^{[3]}_{k,j}(q), \quad q = \mu \text{ or } \nu,
\]

(43)
which is a derivative of the unipolar sigmoid function, which is given as

$$\psi(u_{j,q,k}) = \frac{1}{1+\exp(-\gamma u_{j,q,k})} \quad (44)$$

where $\gamma$ is the positive constant or variable that controls the slope of the sigmoid function.

Let

$$d^{[3]}_{k,j}(\mu, \nu) = 2\gamma \frac{[y^{[3]}_{k,j}(\mu) - y^{[3]}_{k,j}(\nu)]}{S'_{k}(\mu, \nu)} \quad (45)$$

and

$$\delta^{[3]}_{k,j}(q) = d^{[3]}_{k,j}(\mu, \nu)(1 - y^{[3]}_{k,j}(q)) y^{[3]}_{k,j}(q), \quad q = \mu \text{ or } \nu. \quad (46)$$

where $d^{[3]}_{k,j}(\mu, \nu)$ is the change in the output scaled by the normalized interpattern distance change when we present patterns $\mu$ and $\nu$, which both belong to same class $k$. $\delta^{[3]}_{k,j}(\mu)$ and $\delta^{[3]}_{k,j}(\nu)$ will be backpropagated to layer 2. Substituting Eq. (42), Eq. (44) and Eq. (45) into Eq. (41), we get

$$\frac{\partial E_{k,j}(\mu, \nu)}{\partial w^{[3]}_{ij}} = \delta^{[3]}_{k,j}(\mu) y^{[2]}_{k,j}(\mu) - \delta^{[3]}_{k,j}(\nu) y^{[2]}_{k,j}(\nu). \quad (47)$$

The updating rule for the third layer is obtained by substituting Eq. (46) into Eq. (39), which gives

$$w^{[3] \text{new}}_{ij} = w^{[3] \text{old}}_{ij} - \eta (\delta^{[3]}_{k,j}(\mu) y^{[2]}_{k,j}(\mu) - \delta^{[3]}_{k,j}(\nu) y^{[2]}_{k,j}(\nu)). \quad (48)$$

The weights of the second layer of the neural network are updated using the following equation:

$$w^{[2] \text{new}}_{ij} = w^{[2] \text{old}}_{ij} - \eta \left[ \delta^{[2]}_{k,j}(\mu) y^{[1]}_{k,j}(\mu) - \delta^{[2]}_{k,j}(\nu) y^{[1]}_{k,j}(\nu) \right], \quad (49)$$

where

$$\delta^{[2]}_{k,j}(q) = d^{[2]}_{k,j}(q)(1 - y^{[2]}_{k,j}(q)) y^{[2]}_{k,j}(q), \quad q = \mu \text{ or } \nu, \quad (50)$$

where

$$d^{[2]}_{k,j}(q) \equiv \sum_{m=1}^{w} \delta^{[3]}_{k,m}(q) w^{[3]}_{jm}, \quad q = \mu \text{ or } \nu. \quad (51)$$

where $\delta^{[3]}_{k,m}(q) \equiv \gamma^2 d^{[3]}_{k,m}(\mu, \nu)(1 - y^{[3]}_{k,m}(q)) y^{[3]}_{k,m}(q), \quad q = \mu \text{ or } \nu \quad (52)$
The weights of the first layer of the neural network are updated using the following equation:

$$w_{ij}^{[1]new} = w_{ij}^{[1]old} - \eta \left[ \delta_{k,j}^{[1]}(\mu) \gamma_{k,j}^{[1]}(\mu) - \delta_{k,j}^{[1]}(\nu) \gamma_{k,j}^{[1]}(\nu) \right],$$

(53)

where

$$\delta_{k,j}^{[1]}(q) = d_{k,j}^{[1]}(q)(1 - y_{k,j}^{[1]}(q))y_{k,j}^{[1]}(q), \quad q = \mu \text{ or } \nu,$$

(54)

$$d_{k,j}^{[1]}(q) = \sum_{n=1}^{m} \delta_{k,n}^{[2]}(q)w_{mn}^{[2]}, \quad q = \mu \text{ or } \nu,$$

(55)

and

$$\delta_{k,n}^{[2]}(q) = \gamma^2 d_{k,n}^{[2]}(q)(1 - y_{k,n}^{[2]}(q))y_{k,n}^{[2]}(q), \quad q = \mu \text{ or } \nu.$$

(56)

The proposed algorithm for feature extraction of handwritten characters is given as follows:

1. Define the number of classes = 7.
2. Define the number of handwritten patterns used for each character = 50.
3. Read all the 50 × 7 patterns one by one and store them.
4. Present the patterns one by one to the competitive learning neural network (CLN), in which the number of output neurons is equal to the number of classes and perform following steps for each pattern,
   a) Calculate $O_j$ for each pattern.
   b) Pick out the winning neuron by using the lateral weights of the neural network.
   c) Award the winning neuron for the given input pattern.
   d) The winning neuron will represent the class of the given pattern.
5. Now use a feedforward neural network with the proposed Sammon’s stress to perform feature extraction. Create a feedforward neural network with 1024 inputs, 64 neurons in the first hidden layer, 32 neurons in the second hidden layer and 16 outputs.
6. Initialize all the weights of neural network randomly.
7. Find $S(\mu, \nu)$
8. Run the neural network for 100 iterations or until it stops when the modified Sammon’s stress is reaches to a particular value per iteration.
   a) Select a pair of pattern $(\mu, \nu)$ such that $\mu$ goes from 1 to $n - 1$ and $\nu$ goes from $\mu + 1$ to 1, where $n$ is the total number of patterns.
   i. Present the first pattern $\mu$ to the network and find the output.
   ii. Present the second pattern $\nu$ to the network and find the output.
   iii. Find $S(\mu, \nu)$.
   iv. Calculate the error $E_\epsilon(\mu, \nu)$ if patterns $\mu$ and $\nu$ belong to the same classes; otherwise, calculate the error $E_\epsilon(\mu, \nu)$, which means both the two patterns belong to
different classes.

v. Update the weights of the neural network, using the errors calculated in step iv.

9. Once the feedforward neural network is trained, save the weights of the neural network.

10. Now present any pattern which belongs to any class for which the network is trained. It will produce the features of the pattern.

3. PRINCIPAL COMPONENT ANALYSIS NEURAL NETWORK

The Principal Component Analysis (PCA) neural network architecture proposed by Rubner is one of the unsupervised linear models that require fewer computations, and it is used for comparison [19, 27]. The neural network contains \( d \) inputs and \( m \) outputs. Each input node \( i \) is connected to each output unit \( j \) with connection strength \( \xi_{ij} \). Therefore,

\[
o_j = \sum_{i=1}^{n} \xi_{ji} x_i + \sum_{l=1}^{k} \xi_{jl} y_l,
\]

where \( \{x_k \mid k = 1, 2, \ldots, n\} \) is a set of \( n \) input vectors with zero mean, and \( \{o_k \mid k = 1, 2, \ldots, n\} \) be their corresponding output vectors produced by the networks. The weights on connections between the input nodes and the output layer are adjusted upon presentation of an input vector \( x_k = (x_{k1}, x_{k2}, \ldots, x_{km}) \), according to the Hebbian rule, i.e.

\[
\Delta \xi_{ji} = \eta x_k o_{kj} \quad i, j = 1, 2, \ldots, d,
\]

where \( \eta \) is the learning rate. The lateral synaptic weights, on the other hand, are updated according to the anti-Hebbian rule as

\[
\Delta \xi_{jl} = \mu o_{kl} o_{kj} \quad l < j,
\]

where \( \mu \) is another positive learning rate and should fulfill the following condition:

\[
\frac{\eta(\lambda_1 - \lambda)}{\lambda_1(1 + \eta\lambda_d)} < \mu < \frac{2}{\lambda_1}.
\]

Then all the lateral weights will rapidly vanish and the network will converge to a state in which the \( d \) output vectors are \( d \) eigenvectors of the covariance matrix of the input patterns with the eigenvalues \( \lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_d \).

4. RESULTS AND DISCUSSION

The proposed neural network was tested on a data set consisting of three hundred and fifty handwritten characters which belonged to seven different classes of charac-
ters “A”, “B”, “C”, “D”, “E”, “H” and “J”. A sample data set of such characters is shown in Fig. 3. All patterns used had the same dimensionality but different characteristics. The performance of the proposed neural networks was evaluated based on the visual judgment of features and some numerical criteria as well.

We compared the results obtained using our method, the Modified Sammon’s method, with those obtained using two other methods: the PCA method, which uses Artificial Neural Networks, and Sammon’s method, which also uses Artificial Neural Networks along with the Sammon’s stress. Our method also uses Artificial Neural Networks but with the modified Sammon’s stress. The criteria used were firstly, the normalized classification error rate ratio \( R_e \) and secondly the minimum level of the modified Sammon’s stress \( E \) achieved by the three methods. The Sammon’s stress measured how well the extracted features preserved inter pattern distances, while \( R_e \) measured how good the extracted features were in terms of classification accuracy. The normalized classification error rate ratio \( R_e \) was defined as follows:

\[
R_e = \frac{1 + P_e(\text{projected})}{1 + P_e(\text{original})},
\]

(61)

where \( P_e(\text{original}) \) is the nearest neighbor classification error rate of the original data and \( P_e(\text{projected}) \) is the nearest neighbor classification error rate of the projected data. The value of \( R_e \) was within the range 0.5 – 2.0. When \( R_e = 1.0 \), this meant that the extracted features have same discriminatory power as the original data for a given classifier (the nearest neighbor classifier in our case). If \( R_e < 1.0 \), the extracted features had better performance in terms of classification accuracy. If \( R_e > 1.0 \), the extracted features had lost...
some of the critical information required for classification accuracy. Therefore, the extracted features could be considered poor features. The results obtained using the neural network with the proposed Sammon’s stress were compared with the results obtained using the PCA neural network and the unsupervised feedforward neural network with the standard Sammon’s stress.

Table 1. (a) A sample set of feature vectors of hand written characters used to train the PCA neural network.

<table>
<thead>
<tr>
<th>Feature No.</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>H</th>
<th>J</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.528251</td>
<td>0.792037</td>
<td>0.224153</td>
<td>0.408848</td>
<td>0.32402</td>
<td>0.362553</td>
<td>0.096117</td>
</tr>
<tr>
<td>2</td>
<td>0.591185</td>
<td>0.813644</td>
<td>0.267894</td>
<td>0.441904</td>
<td>0.387404</td>
<td>0.403874</td>
<td>0.080849</td>
</tr>
<tr>
<td>3</td>
<td>0.629202</td>
<td>0.862013</td>
<td>0.29974</td>
<td>0.473129</td>
<td>0.412869</td>
<td>0.435124</td>
<td>0.069787</td>
</tr>
<tr>
<td>4</td>
<td>0.606613</td>
<td>0.883183</td>
<td>0.305043</td>
<td>0.508776</td>
<td>0.391858</td>
<td>0.443299</td>
<td>0.124002</td>
</tr>
<tr>
<td>5</td>
<td>0.595954</td>
<td>0.872145</td>
<td>0.280326</td>
<td>0.445019</td>
<td>0.351249</td>
<td>0.404677</td>
<td>0.109465</td>
</tr>
<tr>
<td>6</td>
<td>0.613231</td>
<td>0.846216</td>
<td>0.247539</td>
<td>0.452893</td>
<td>0.388605</td>
<td>0.401966</td>
<td>0.093244</td>
</tr>
<tr>
<td>7</td>
<td>0.589814</td>
<td>0.892751</td>
<td>0.313716</td>
<td>0.464757</td>
<td>0.387158</td>
<td>0.425039</td>
<td>0.107354</td>
</tr>
<tr>
<td>8</td>
<td>0.597833</td>
<td>0.836441</td>
<td>0.275122</td>
<td>0.49467</td>
<td>0.38368</td>
<td>0.40056</td>
<td>0.098986</td>
</tr>
<tr>
<td>9</td>
<td>0.591649</td>
<td>0.846947</td>
<td>0.27211</td>
<td>0.455021</td>
<td>0.404677</td>
<td>0.097485</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>0.597809</td>
<td>0.888383</td>
<td>0.274777</td>
<td>0.457523</td>
<td>0.386447</td>
<td>0.429758</td>
<td>0.105265</td>
</tr>
<tr>
<td>11</td>
<td>0.599313</td>
<td>0.882731</td>
<td>0.288251</td>
<td>0.449746</td>
<td>0.384196</td>
<td>0.40235</td>
<td>0.11447</td>
</tr>
<tr>
<td>12</td>
<td>0.597956</td>
<td>0.852047</td>
<td>0.262887</td>
<td>0.478207</td>
<td>0.38621</td>
<td>0.409284</td>
<td>0.101702</td>
</tr>
<tr>
<td>13</td>
<td>0.597809</td>
<td>0.888383</td>
<td>0.274777</td>
<td>0.457523</td>
<td>0.386447</td>
<td>0.429758</td>
<td>0.105265</td>
</tr>
</tbody>
</table>

Table 1. (b) A sample set of feature vectors of hand written characters used to train the Sammon’s stress neural network.

<table>
<thead>
<tr>
<th>Feature No.</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>H</th>
<th>J</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.394283</td>
<td>0.745599</td>
<td>0.186678</td>
<td>0.401543</td>
<td>0.332678</td>
<td>0.359931</td>
<td>0.171959</td>
</tr>
<tr>
<td>2</td>
<td>0.404512</td>
<td>0.75972</td>
<td>0.194763</td>
<td>0.411906</td>
<td>0.342057</td>
<td>0.369563</td>
<td>0.179878</td>
</tr>
<tr>
<td>3</td>
<td>0.25034</td>
<td>0.5482</td>
<td>0.074456</td>
<td>0.256653</td>
<td>0.198333</td>
<td>0.405474</td>
<td>0.136848</td>
</tr>
<tr>
<td>4</td>
<td>0.310986</td>
<td>0.631467</td>
<td>0.121802</td>
<td>0.317786</td>
<td>0.254935</td>
<td>0.279786</td>
<td>0.105265</td>
</tr>
<tr>
<td>5</td>
<td>0.370427</td>
<td>0.712691</td>
<td>0.167791</td>
<td>0.37732</td>
<td>0.325935</td>
<td>0.187986</td>
<td>0.090677</td>
</tr>
<tr>
<td>6</td>
<td>0.379356</td>
<td>0.724995</td>
<td>0.174903</td>
<td>0.386519</td>
<td>0.318753</td>
<td>0.435363</td>
<td>0.096777</td>
</tr>
<tr>
<td>7</td>
<td>0.498112</td>
<td>0.887979</td>
<td>0.267255</td>
<td>0.506228</td>
<td>0.429432</td>
<td>0.459926</td>
<td>0.251221</td>
</tr>
<tr>
<td>8</td>
<td>0.517103</td>
<td>0.914036</td>
<td>0.282312</td>
<td>0.525526</td>
<td>0.447119</td>
<td>0.478158</td>
<td>0.269996</td>
</tr>
<tr>
<td>9</td>
<td>0.452681</td>
<td>0.825866</td>
<td>0.231999</td>
<td>0.46053</td>
<td>0.387173</td>
<td>0.41606</td>
<td>0.217617</td>
</tr>
<tr>
<td>10</td>
<td>0.493355</td>
<td>0.881504</td>
<td>0.264051</td>
<td>0.501757</td>
<td>0.425274</td>
<td>0.455179</td>
<td>0.247985</td>
</tr>
<tr>
<td>11</td>
<td>0.346308</td>
<td>0.679437</td>
<td>0.149264</td>
<td>0.353187</td>
<td>0.287413</td>
<td>0.313508</td>
<td>0.131562</td>
</tr>
<tr>
<td>12</td>
<td>0.421979</td>
<td>0.783727</td>
<td>0.208355</td>
<td>0.424987</td>
<td>0.358746</td>
<td>0.386675</td>
<td>0.193379</td>
</tr>
<tr>
<td>13</td>
<td>0.577582</td>
<td>0.997653</td>
<td>0.330073</td>
<td>0.586718</td>
<td>0.504167</td>
<td>0.536603</td>
<td>0.312465</td>
</tr>
<tr>
<td>14</td>
<td>0.262293</td>
<td>0.564336</td>
<td>0.083857</td>
<td>0.268629</td>
<td>0.209381</td>
<td>0.23254</td>
<td>0.071398</td>
</tr>
<tr>
<td>15</td>
<td>0.54212</td>
<td>0.948269</td>
<td>0.301846</td>
<td>0.550266</td>
<td>0.470485</td>
<td>0.501962</td>
<td>0.284921</td>
</tr>
<tr>
<td>16</td>
<td>0.356858</td>
<td>0.694178</td>
<td>0.157358</td>
<td>0.363887</td>
<td>0.297604</td>
<td>0.323818</td>
<td>0.143577</td>
</tr>
</tbody>
</table>
Table 1. (c) A sample set of feature vectors of handwritten character used to train the modified Sammon's stress neural network.

<table>
<thead>
<tr>
<th>Feature No.</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>H</th>
<th>J</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.396926</td>
<td>0.761655</td>
<td>0.256116</td>
<td>0.407215</td>
<td>0.334488</td>
<td>0.3663</td>
<td>0.171561</td>
</tr>
<tr>
<td>2</td>
<td>0.293872</td>
<td>0.619366</td>
<td>0.168143</td>
<td>0.302524</td>
<td>0.237459</td>
<td>0.266517</td>
<td>0.092584</td>
</tr>
<tr>
<td>3</td>
<td>0.359202</td>
<td>0.709795</td>
<td>0.224085</td>
<td>0.36884</td>
<td>0.297899</td>
<td>0.258761</td>
<td>0.115751</td>
</tr>
<tr>
<td>4</td>
<td>0.324125</td>
<td>0.661371</td>
<td>0.194068</td>
<td>0.333165</td>
<td>0.26576</td>
<td>0.295861</td>
<td>0.11751</td>
</tr>
<tr>
<td>5</td>
<td>0.34277</td>
<td>0.687519</td>
<td>0.209999</td>
<td>0.351953</td>
<td>0.283442</td>
<td>0.314185</td>
<td>0.130165</td>
</tr>
<tr>
<td>6</td>
<td>0.524182</td>
<td>0.938086</td>
<td>0.365061</td>
<td>0.535587</td>
<td>0.453057</td>
<td>0.489935</td>
<td>0.268846</td>
</tr>
<tr>
<td>7</td>
<td>0.313061</td>
<td>0.646438</td>
<td>0.18432</td>
<td>0.322828</td>
<td>0.25568</td>
<td>0.285344</td>
<td>0.1075</td>
</tr>
<tr>
<td>8</td>
<td>0.33708</td>
<td>0.674688</td>
<td>0.202396</td>
<td>0.343129</td>
<td>0.274916</td>
<td>0.305466</td>
<td>0.123257</td>
</tr>
<tr>
<td>9</td>
<td>0.437632</td>
<td>0.818204</td>
<td>0.291042</td>
<td>0.447984</td>
<td>0.371918</td>
<td>0.405864</td>
<td>0.202621</td>
</tr>
<tr>
<td>10</td>
<td>0.282186</td>
<td>0.603635</td>
<td>0.158461</td>
<td>0.290989</td>
<td>0.226823</td>
<td>0.255549</td>
<td>0.083694</td>
</tr>
<tr>
<td>11</td>
<td>0.39614</td>
<td>0.76074</td>
<td>0.255536</td>
<td>0.406016</td>
<td>0.33208</td>
<td>0.365804</td>
<td>0.170771</td>
</tr>
<tr>
<td>12</td>
<td>0.37821</td>
<td>0.734313</td>
<td>0.239358</td>
<td>0.38703</td>
<td>0.31571</td>
<td>0.347816</td>
<td>0.156275</td>
</tr>
<tr>
<td>13</td>
<td>0.378652</td>
<td>0.736962</td>
<td>0.241008</td>
<td>0.38644</td>
<td>0.31687</td>
<td>0.348943</td>
<td>0.157704</td>
</tr>
<tr>
<td>14</td>
<td>0.56231</td>
<td>0.990396</td>
<td>0.39756</td>
<td>0.574078</td>
<td>0.488599</td>
<td>0.526961</td>
<td>0.298129</td>
</tr>
<tr>
<td>15</td>
<td>0.247969</td>
<td>0.556908</td>
<td>0.12964</td>
<td>0.256984</td>
<td>0.195238</td>
<td>0.222407</td>
<td>0.05788</td>
</tr>
<tr>
<td>16</td>
<td>0.396926</td>
<td>0.761655</td>
<td>0.256116</td>
<td>0.407215</td>
<td>0.334488</td>
<td>0.3663</td>
<td>0.171561</td>
</tr>
</tbody>
</table>

Table 2. The normalized classification error rate \( R_e \) and Sammon’s stress \( E \) of the neural networks after training.

<table>
<thead>
<tr>
<th>Criteria</th>
<th>PCA Method</th>
<th>Sammon’s Method</th>
<th>Modified Sammon’s Method</th>
<th>( \rho = 1 )</th>
<th>( \rho = 0.8 )</th>
<th>( \rho = 0.5 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( R_e )</td>
<td>0.9090</td>
<td>0.8181</td>
<td>0.7721</td>
<td>0.8181</td>
<td>0.7271</td>
<td></td>
</tr>
<tr>
<td>( E )</td>
<td>0.90405</td>
<td>0.4667</td>
<td>0.3765</td>
<td>0.8707</td>
<td>0.5395</td>
<td></td>
</tr>
</tbody>
</table>

Tables 1 (a), (b), and (c) show the sample feature vectors of each character obtained using the PCA network, the Sammon’s Stress neural network, and the Modified Sammon’s Stress neural network, respectively. The features of each character consisted of sixteen numerical values. Table 2 shows the normalized classification error rate and Sammon’s perform feature extraction. It also contains the results obtained using the proposed neural method for different values of \( \rho \), i.e., 1, 0.8 and 0.5. We observe that when \( \rho = 0.8 \), the results deteriorated a little bit, but that when \( \rho = 0.5 \), the Sammon’s stress and normalized classification error rate were low as compared to the results obtained when \( \rho = 1, 0.8 \).

Tables 3 (a), (b), and (c) show feature vectors of the handwritten characters used as testing data for the PCA neural network, Sammon’s Stress neural network and Modified Sammon’s Stress neural network, respectively. Table 4 shows the Sammon’s stress and normalized classification error rate of the noisy test data set shown in Fig. 4. It shows that the Sammon’s stress and normalized classification error rate achieved using our proposed neural network were lower than those obtained using the other two networks.
Fig. 4. A sample of degraded images used to test the proposed network.

Table 3. (a) A sample set of feature vectors of handwritten characters used to test the PCA neural network.

<table>
<thead>
<tr>
<th>Feature No.</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>H</th>
<th>J</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.331403</td>
<td>0.865523</td>
<td>0.264218</td>
<td>0.398271</td>
<td>0.335991</td>
<td>0.24557</td>
<td>0.072839</td>
</tr>
<tr>
<td>2</td>
<td>0.347022</td>
<td>0.882851</td>
<td>0.269012</td>
<td>0.428511</td>
<td>0.399547</td>
<td>0.321</td>
<td>0.079478</td>
</tr>
<tr>
<td>3</td>
<td>0.39623</td>
<td>0.966118</td>
<td>0.349497</td>
<td>0.437088</td>
<td>0.440808</td>
<td>0.329237</td>
<td>0.116048</td>
</tr>
<tr>
<td>4</td>
<td>0.437547</td>
<td>0.977181</td>
<td>0.328845</td>
<td>0.517829</td>
<td>0.431759</td>
<td>0.348935</td>
<td>0.113889</td>
</tr>
<tr>
<td>5</td>
<td>0.401209</td>
<td>0.951921</td>
<td>0.306036</td>
<td>0.439936</td>
<td>0.398356</td>
<td>0.319266</td>
<td>0.094709</td>
</tr>
<tr>
<td>6</td>
<td>0.393415</td>
<td>0.929103</td>
<td>0.274334</td>
<td>0.470099</td>
<td>0.402337</td>
<td>0.298407</td>
<td>0.116501</td>
</tr>
<tr>
<td>7</td>
<td>0.374366</td>
<td>0.954326</td>
<td>0.358254</td>
<td>0.472146</td>
<td>0.427144</td>
<td>0.338264</td>
<td>0.10759</td>
</tr>
<tr>
<td>8</td>
<td>0.387887</td>
<td>0.901248</td>
<td>0.307156</td>
<td>0.432423</td>
<td>0.403763</td>
<td>0.277729</td>
<td>0.120733</td>
</tr>
<tr>
<td>9</td>
<td>0.336651</td>
<td>0.967424</td>
<td>0.285954</td>
<td>0.489583</td>
<td>0.437428</td>
<td>0.322503</td>
<td>0.086336</td>
</tr>
<tr>
<td>10</td>
<td>0.413389</td>
<td>0.985261</td>
<td>0.30957</td>
<td>0.470064</td>
<td>0.431906</td>
<td>0.335119</td>
<td>0.16055</td>
</tr>
<tr>
<td>11</td>
<td>0.41366</td>
<td>1</td>
<td>0.335278</td>
<td>0.481051</td>
<td>0.403714</td>
<td>0.320806</td>
<td>0.124183</td>
</tr>
<tr>
<td>12</td>
<td>0.343547</td>
<td>0.910246</td>
<td>0.28191</td>
<td>0.459821</td>
<td>0.440312</td>
<td>0.297574</td>
<td>0.111745</td>
</tr>
<tr>
<td>13</td>
<td>0.375707</td>
<td>0.929592</td>
<td>0.385119</td>
<td>0.441441</td>
<td>0.442394</td>
<td>0.299147</td>
<td>0.167508</td>
</tr>
<tr>
<td>14</td>
<td>0.362499</td>
<td>0.930276</td>
<td>0.301251</td>
<td>0.422282</td>
<td>0.364375</td>
<td>0.303587</td>
<td>0.125135</td>
</tr>
<tr>
<td>15</td>
<td>0.385277</td>
<td>0.967699</td>
<td>0.377693</td>
<td>0.450886</td>
<td>0.437667</td>
<td>0.329806</td>
<td>0.121799</td>
</tr>
<tr>
<td>16</td>
<td>0.356236</td>
<td>0.949829</td>
<td>0.299815</td>
<td>0.436705</td>
<td>0.446882</td>
<td>0.312975</td>
<td>0.133435</td>
</tr>
</tbody>
</table>
Table 3. (b) A sample set of feature vectors of hand written character used to test the Sammon’s stress neural network.

<table>
<thead>
<tr>
<th>Feature No.</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>H</th>
<th>J</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.357711</td>
<td>0.740554</td>
<td>0.253641</td>
<td>0.402856</td>
<td>0.347918</td>
<td>0.28159</td>
<td>0.118922</td>
</tr>
<tr>
<td>2</td>
<td>0.367565</td>
<td>0.754851</td>
<td>0.262489</td>
<td>0.413445</td>
<td>0.357848</td>
<td>0.29891</td>
<td>0.126378</td>
</tr>
<tr>
<td>3</td>
<td>0.215731</td>
<td>0.540122</td>
<td>0.127872</td>
<td>0.254248</td>
<td>0.207843</td>
<td>0.150903</td>
<td>0.013511</td>
</tr>
<tr>
<td>4</td>
<td>0.275589</td>
<td>0.624883</td>
<td>0.180976</td>
<td>0.316949</td>
<td>0.266604</td>
<td>0.205773</td>
<td>0.058171</td>
</tr>
<tr>
<td>5</td>
<td>0.334302</td>
<td>0.707227</td>
<td>0.232695</td>
<td>0.377919</td>
<td>0.324556</td>
<td>0.259845</td>
<td>0.101636</td>
</tr>
<tr>
<td>6</td>
<td>0.342923</td>
<td>0.719337</td>
<td>0.240238</td>
<td>0.387678</td>
<td>0.333293</td>
<td>0.267587</td>
<td>0.107593</td>
</tr>
<tr>
<td>7</td>
<td>0.460217</td>
<td>0.884122</td>
<td>0.344158</td>
<td>0.510334</td>
<td>0.448771</td>
<td>0.37525</td>
<td>0.195155</td>
</tr>
<tr>
<td>8</td>
<td>0.478725</td>
<td>0.910604</td>
<td>0.361031</td>
<td>0.529667</td>
<td>0.467202</td>
<td>0.391985</td>
<td>0.208929</td>
</tr>
<tr>
<td>9</td>
<td>0.419006</td>
<td>0.821637</td>
<td>0.304558</td>
<td>0.505413</td>
<td>0.444242</td>
<td>0.370573</td>
<td>0.191932</td>
</tr>
<tr>
<td>10</td>
<td>0.455076</td>
<td>0.877801</td>
<td>0.311171</td>
<td>0.353365</td>
<td>0.301184</td>
<td>0.237628</td>
<td>0.083668</td>
</tr>
<tr>
<td>11</td>
<td>0.384787</td>
<td>0.79936</td>
<td>0.277883</td>
<td>0.432127</td>
<td>0.37441</td>
<td>0.306305</td>
<td>0.139252</td>
</tr>
<tr>
<td>12</td>
<td>0.537871</td>
<td>0.995449</td>
<td>0.414204</td>
<td>0.592612</td>
<td>0.5262</td>
<td>0.446749</td>
<td>0.253136</td>
</tr>
<tr>
<td>13</td>
<td>0.227532</td>
<td>0.556117</td>
<td>0.13823</td>
<td>0.266769</td>
<td>0.219431</td>
<td>0.161785</td>
<td>0.022874</td>
</tr>
<tr>
<td>14</td>
<td>0.503092</td>
<td>0.945346</td>
<td>0.382513</td>
<td>0.554971</td>
<td>0.491826</td>
<td>0.414405</td>
<td>0.226724</td>
</tr>
<tr>
<td>15</td>
<td>0.320398</td>
<td>0.687957</td>
<td>0.220796</td>
<td>0.364213</td>
<td>0.311389</td>
<td>0.247362</td>
<td>0.092149</td>
</tr>
</tbody>
</table>

Table 3. (c) A sample set of feature vectors of hand written character used to test the modified Sammon’s stress neural network.

<table>
<thead>
<tr>
<th>Feature No.</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>H</th>
<th>J</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.367291</td>
<td>0.751966</td>
<td>0.253641</td>
<td>0.402856</td>
<td>0.347918</td>
<td>0.28159</td>
<td>0.118922</td>
</tr>
<tr>
<td>2</td>
<td>0.265794</td>
<td>0.609436</td>
<td>0.169104</td>
<td>0.299393</td>
<td>0.247378</td>
<td>0.136551</td>
<td>0.046664</td>
</tr>
<tr>
<td>3</td>
<td>0.22776</td>
<td>0.556081</td>
<td>0.135945</td>
<td>0.260167</td>
<td>0.210766</td>
<td>0.104449</td>
<td>0.019091</td>
</tr>
<tr>
<td>4</td>
<td>0.330018</td>
<td>0.699874</td>
<td>0.226618</td>
<td>0.366762</td>
<td>0.310699</td>
<td>0.191081</td>
<td>0.094531</td>
</tr>
<tr>
<td>5</td>
<td>0.295221</td>
<td>0.651539</td>
<td>0.195765</td>
<td>0.330568</td>
<td>0.27664</td>
<td>0.161721</td>
<td>0.06824</td>
</tr>
<tr>
<td>6</td>
<td>0.313955</td>
<td>0.677927</td>
<td>0.211831</td>
<td>0.349702</td>
<td>0.295028</td>
<td>0.177544</td>
<td>0.082415</td>
</tr>
<tr>
<td>7</td>
<td>0.492484</td>
<td>0.929134</td>
<td>0.370052</td>
<td>0.535644</td>
<td>0.469707</td>
<td>0.328568</td>
<td>0.214068</td>
</tr>
<tr>
<td>8</td>
<td>0.284948</td>
<td>0.636542</td>
<td>0.186089</td>
<td>0.319493</td>
<td>0.26635</td>
<td>0.152701</td>
<td>0.060704</td>
</tr>
<tr>
<td>9</td>
<td>0.305023</td>
<td>0.665024</td>
<td>0.20436</td>
<td>0.34029</td>
<td>0.286127</td>
<td>0.169889</td>
<td>0.075968</td>
</tr>
<tr>
<td>10</td>
<td>0.415006</td>
<td>0.594122</td>
<td>0.159339</td>
<td>0.287873</td>
<td>0.236432</td>
<td>0.127097</td>
<td>0.03812</td>
</tr>
<tr>
<td>11</td>
<td>0.320398</td>
<td>0.546191</td>
<td>0.130293</td>
<td>0.2536</td>
<td>0.204371</td>
<td>0.098601</td>
<td>0.013696</td>
</tr>
</tbody>
</table>
Table 4. The normalized classification error rate $R_e$ and Sammon’s stress $E$ of the noisy test data.

<table>
<thead>
<tr>
<th>Criteria</th>
<th>PCA Method</th>
<th>Sammon’s Method</th>
<th>Modified Sammon’s Method $(\rho = 1)$</th>
<th>$(\rho = 0.8)$</th>
<th>$(\rho = 0.5)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$R_e$</td>
<td>0.8181</td>
<td>0.8181</td>
<td>0.8090</td>
<td>0.833</td>
<td>0.666</td>
</tr>
<tr>
<td>$E$</td>
<td>0.5635</td>
<td>0.9307</td>
<td>0.7213</td>
<td>0.8912</td>
<td>0.0978</td>
</tr>
</tbody>
</table>

5. CONCLUSIONS

A neural network model has been proposed for the robust feature extraction of hand written characters using the modified Sammon’s stress for learning purposes. This model can be applied to feature extraction problems of degraded and omni font characters as well. The method consists of two parts. In the first part, we estimate rough classes using the unsupervised Competitive Learning Network and in the second part, we use a feedforward neural network to perform feature extraction based on the modified Sammon’s stress using partial information obtained from the Competitive Learning Network. These features are compared with the features obtained using the PCA neural network and neural network with standard Sammon’s stress. Their performance was evaluated in terms of the Sammon’s stress and classification accuracy. The performance of the networks showed that the features of the proposed modified Sammon’s stress neural network achieve better performance than those of the PCA neural network and neural network with Sammon’s stress.

REFERENCES

Abdul Jalil was born in Attock, Pakistan in 1961. He received the M.S. degree from Quaid-i-Azam University, Islamabad in 1986 and the M.Phil. degree from the same University in 2000, both in Electronics. Since 2000, he has been perusing Ph.D. degree in Computer Engineering, with the School of Engineering and Science, M. A. Jinnah University, Islamabad. His research interests are image processing and pattern recognition.

Ijaz Mansoor Qureshi was born in Gujrat, Pakistan in 1954. He received the M.S. degree from Middle East Tech. University, Ankara in 1980 and Ph.D. degree in High Energy Physics from University of Toronto. Currently, he is working as Professor in School of Engineering and Sciences, M. A. Jinnah University, Islamabad. His research interests include image processing, pattern recognition, image compression, image restoration, biometrics and blind channel equalization.

Tanweer Ahmad Cheema was born in Gujrat, Pakistan in 1973. He received the M.S. degree from Agricultural University, Faisalabad in 1997, in Physics. He received M.Phil. degree from Quaid-i-Azam University, Islamabad in 2000, in Electronics. He is currently working as Assistant Professor in School of Engineering and Sciences, M. A. Jinnah University, Islamabad. His research interests include image processing, image restoration, image compression, and image recognition.

Aqdas Naveed was born in Chakwal, Pakistan in 1967. He received his M.S. degree from University of Bahawalpure in Physics. He received M.Phil. degree from Quaid-i-Azam University, Islamabad in 2000, in Electronics. He is currently working as Assistant Professor in School of Engineering and Sciences, M. A. Jinnah University, Islamabad. His research interests include blind channel equilization, image processing, and neural networks.