PERGAMON

Pattern Recognition 33 (2000) 945–960

PATTERN RECOGNITION
THE JOURNAL OF THE PATTERN RECOGNITION SOCIETY

www.elsevier.com/locate/patcog

# Neural networks with hybrid morphological/rank/linear nodes: a unifying framework with applications to handwritten character recognition☆

Lúcio F.C. Pessoa[a,*], Petros Maragos[b]

[a]Motorola, Inc., 3501 Ed Bluestein Blvd., MD: TX11/H4, Austin, TX 78721, USA
[b]Department of Electrical and Computer Engineering, National Technical University of Athens, Zografou 15773, Athens, Greece

Received 15 December 1998; received in revised form 25 March 1999; accepted 7 April 1999

## Abstract

In this paper, the general class of morphological/rank/linear (MRL) multilayer feed-forward neural networks (NNs) is presented as a unifying signal processing tool that incorporates the properties of multilayer perceptrons (MLPs) and morphological/rank neural networks (MRNNs). The fundamental processing unit of MRL-NNs is the MRL-filter, where the combination of inputs in every node is formed by hybrid linear and nonlinear (of the morphological/rank type) operations. For its design we formulate a methodology using ideas from the back-propagation algorithm and robust techniques to circumvent the non-differentiability of rank functions. Extensive experimental results are presented from the problem of handwritten character recognition, which suggest that MRL-NNs not only provide better or similar performance when compared to MLPs but also can be trained faster. The MRL-NNs are a broad interesting class of nonlinear systems with many promising applications in pattern recognition and signal/image processing. © 2000 Pattern Recognition Society. Published by Elsevier Science Ltd. All rights reserved.

*Keywords:* Morphological systems; MRL-filters; Neural networks; Back-propagation algorithm; Handwritten character recognition

## 1. Introduction

Multilayer feed-forward neural networks, or simply neural networks (NNs), represent an important class of nonlinear systems widely used in problems of signal/image processing and pattern recognition. Their applications in signal/image processing usually employ networks with a single output, which are sometimes called NN-filters. Furthermore, adaptive filters and NNs are closely related, so that their adaptation/training can be studied under the same framework [1]. In this sense, the design of an NN-filter corresponds to the training process of its embedded NN. The usefulness of NNs can be efficiently investigated due to the existence of the back-propagation algorithm [2], which represents a generalization of the LMS algorithm for feed-forward networks. In this way, the system design is viewed as a problem of unconstrained optimization that is iteratively solved by the method of steepest descent.

The node structure in an NN is supposed to model the input–output characteristic of a neuron, and so it represents the essence of the system. The perceptron, i.e., a linear combiner followed by a nonlinearity of the logistic type, is the classic node structure used in NNs. However, it has been observed that logic operations, which are not well modeled by perceptrons, can be generated by some internal interactions in a neuron [3]. For the sake of

☆This work was done while both authors were with the School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, USA. It was supported by the US National Science Foundation under grant MIP–94-21677, and in part by CNPq (Conselho Nacional de Desenvolvimento Científico e Tecnológico), Brasília, Brazil, through a Doctoral Fellowship under grant 200.846/92-2.

*Corresponding author. Tel.: + 1-512-934-6613; fax: + 1-934-6688.

*E-mail addresses:* Lucio.pessoa@motorola.com (L.F.C. Pessoa), Petros.Maragos@cs.ntua.gr (P. Maragos) a better representation of these internal properties, a possible improvement to the basic perceptron model is presented in this paper. We propose the MRL-NNs [4], a general class of NNs where the combination of inputs in every node is formed by hybrid linear and nonlinear (of the morphological/rank type) operations. The fundamental processing unit of this class of systems is the MRL-filter [5], which is a linear combination between a morphological/rank filter and a linear FIR filter. The MRL-NNs have the unifying property that the characteristics of both multilayer perceptrons (MLPs) and morphological/rank neural networks (MRNNs) [6] are observed in the same system. An important special case of MRNNs is the class of min–max classifiers [7], which can provide classification results comparable to MLPs, but with faster training processes. Other related works with min–max operations in networks have appeared in Refs. [8,9]. We show in this paper that the MRL-NNs can solve the parity problem in closed form with about half of the number of nodes usually required by MLPs and a smaller computational complexity. Examples from simple pattern classification problems are also included to provide geometrical insights. These demonstrate the potential of this new structure that offers efficient solutions to pattern classification problems by requiring fewer nodes or fewer parameters to estimate than those needed by MLPs. Next, we formulate a simple and systematic training procedure using ideas from the back-propagation algorithm and robust techniques to circumvent the nondifferentiability of rank functions. Our approach to train the morphological/rank nodes is a theoretically and numerically improved version of the method proposed by Salembier [10,11] to design morphological/rank filters. Finally, we apply the proposed design methodology to problems of optical character recognition and provide extensive experimental evidence showing not only that the MRL-NNs can generate similar or better results when compared with the classical MLPs, but they also usually require less processing time for training.

## 2. The MRL-NN

In general terms, a (multilayer feed-forward) NN is a layered system composed of similar nodes, with some of them nonobservable (hidden), where the node inputs in a given layer depend only on the node outputs from the preceding layer. In addition, no feedback is allowed in the topology of this class of systems. Every node performs a generic composite operation, where an input to the node is first processed by some function $h(\cdot,\cdot)$ of the input and internal weights and then transformed by an activation function $f(\cdot)$. The node structure is defined by the function $h$. In the case of MLPs, $h$ is a linear combination. The activation function $f$ is usually employed for rescaling purposes. We will consider the special cases where $f$ is the identity or a nonlinearity of the logistic type and denote the corresponding systems as NNs of types I and II, respectively. A general NN is formally defined by the following set of recursive equations:

$$\mathbf{y}^{(l)} \equiv F(\mathbf{z}^{(l)}) = (f(z_1^{(l)}), f(z_2^{(l)}), \ldots, f(z_{N_l}^{(l)})), \quad l = 1, 2, \ldots, L,$$
$$z_n^{(l)} \equiv h(\mathbf{y}^{(l-1)}, \mathbf{w}_n^{(l)}), \quad n = 1, 2, \ldots, N_l, \tag{1}$$

where $l$ is the layer number, and $N_l$ is the number of nodes in layer $l$. The weight vectors $\mathbf{w}_n^{(l)}$ represent the tuning parameters in the system. The structure of the $l$th layer is illustrated in Fig. 1. Besides this, the input and output of the system are

$$\begin{aligned}\mathbf{y}^{(0)} &= \mathbf{x} = (x_1, x_2, \ldots, x_{N_0}) \quad \text{(input)},\\ \mathbf{y}^{(L)} &= \mathbf{y} = (y_1, y_2, \ldots, y_{N_L}) \quad \text{(output)}.\end{aligned} \tag{2}$$

Before we define the MRL-NN, we shall review the concept of its fundamental processing unit: The MRL-filter. Let $\mathbf{x} = (x_1, x_2, \ldots x_n)$ in $\mathbb{R}^n$ represent the input signal and $y$ be the output value from the filter. We use a vector notation to represent the values of the 1D or 2D sampled input signal (after some enumeration of the signal samples) inside an $n$-point moving window. The MRL-filter is defined as the shift-invariant system whose local signal transformation rule $\mathbf{x} \mapsto y$ is given by

$$y \equiv \lambda\alpha + (1-\lambda)\beta,$$
$$\alpha = \mathscr{R}_r(\mathbf{x} + \mathbf{a}) = \mathscr{R}_r(x_1 + a_1, x_2 + a_2, \ldots, x_n + a_n),$$
$$\beta = \mathbf{x} \cdot \mathbf{b}' = x_1 b_1 + x_2 b_2 + \cdots + x_n b_n, \tag{3}$$

where $\lambda \in \mathbb{R}$, $\mathbf{a}, \mathbf{b} \in \mathbb{R}^n$, and ' $'$ ' denotes transposition. $\mathscr{R}_r(\mathbf{t})$ is the $r$th rank function of $\mathbf{t} \in \mathbb{R}^n$. It is evaluated by sorting the components of $\mathbf{t} = (t_1, t_2, \ldots, t_n)$ in decreasing order, $t_{(1)} \geqslant t_{(2)} \geqslant \cdots \geqslant t_{(n)}$, and picking the $r$th element of the sorted list, i.e., $\mathscr{R}_r(\mathbf{t}) \equiv t_{(r)}$, $r = 1, 2, \ldots, n$. The vector $\mathbf{b} = (b_1, b_2, \ldots, b_n)$ corresponds to the coefficients of the

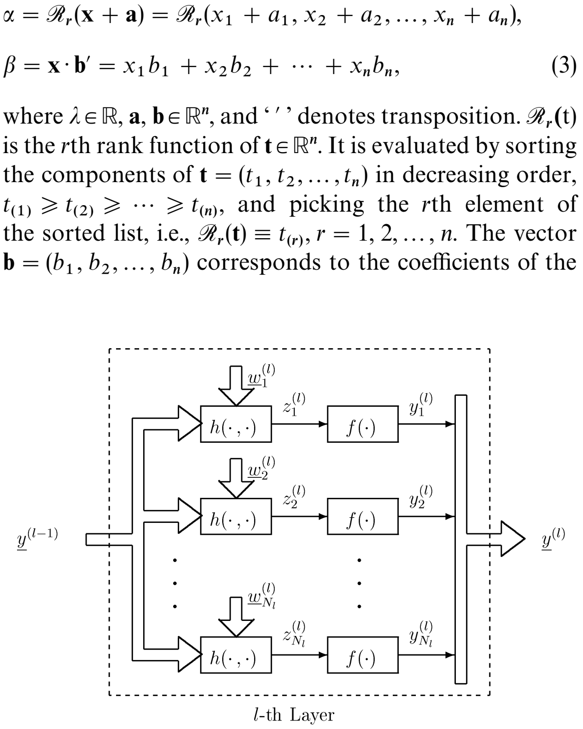


Fig. 1. Structure of the $l$th layer in a general NN.

linear FIR filter, and the vector $\mathbf{a} = (a_1, a_2, \ldots, a_n)$ represents the coefficients of the morphological/rank filter. We call **a** the "structuring element" because for $r = 1$ and $n$ the rank filter becomes the morphological dilation and erosion by a structuring function equal to $\pm\,\mathbf{a}$ within its support. The variables $r$ and $\lambda$ are the rank and mixing parameters of the filter, respectively. If $\lambda \in [0, 1]$, the MRL-filter becomes a convex combination of its components, so that when we increase the contribution of one component, the other one tends to decrease. For every point of the signal, we can easily see from Eq. (3) that we need $2n + 1$ additions, $n + 2$ multiplications and an $n$-point sorting operation.

The MRL-NN is the system defined by Eqs. (1) and (2) such that

$$z_n^{(l)} \equiv \lambda_n^{(l)}\alpha_n^{(l)} + (1 - \lambda_n^{(l)})\beta_n^{(l)},$$

$$\alpha_n^{(l)} = \mathscr{R}_{r_n^{(l)}}(\mathbf{y}^{(l-1)} + \mathbf{a}_n^{(l)}),$$

$$\beta_n^{(l)} = \mathbf{y}^{(l-1)} \cdot (\mathbf{b}_n^{(l)})' + \tau_n^{(l)}, \tag{4}$$

where $\lambda_n^{(l)}, \tau_n^{(l)} \in \mathbb{R}$; $\mathbf{a}_n^{(l)}, \mathbf{b}_n^{(l)} \in \mathbb{R}^{N_{l-1}}$.

Observe from Eqs. (1), (3) and (4) that the underlying function $h$ is an MRL-filter shifted by a threshold $(1 - \lambda_n^{(l)})\tau_n^{(l)}$. The offset variables $\tau_n^{(l)}$ are important when $\lambda_n^{(l)} = 0$. The resulting weight vector for every node is then defined by

$$\mathbf{w}_n^{(l)} \equiv (\mathbf{a}_n^{(l)}, \rho_n^{(l)}, \mathbf{b}_n^{(l)}, \tau_n^{(l)}, \lambda_n^{(l)}), \tag{5}$$

where we use a real variable $\rho_n^{(l)}$ instead of an integer rank variable $r_n^{(l)}$ because we will need to evaluate rank derivatives during the design of MRL-NNs. The relation between $\rho_n^{(l)}$ and $z_n^{(l)}$ will be defined later via a differential equation, and $r_n^{(l)}$ is obtained from $\rho_n^{(l)}$ via the following rescaling:[1]

$$r_n^{(l)} \equiv \left\lfloor N_{l-1} - \frac{N_{l-1} - 1}{1 + \exp(-\rho_n^{(l)})} + 0.5 \right\rfloor, \tag{6}$$

which is a simple way to map from a variable $\rho_n^{(l)} \in \mathbb{R}$ to an integer $r_n^{(l)} \in \{1, 2, \ldots, N_{l-1}\}$. For example, if $\rho_n^{(l)} \to -\infty$, then $r_n^{(l)} \to N_{l-1}$, corresponding to a minimum operation; if $\rho_n^{(l)} \to \infty$, then $r_n^{(l)} \to 1$, corresponding to a maximum operation; if $\rho_n^{(l)} = 0$, then $r_n^{(l)} = \lfloor N_{l-1}/2 + 1 \rfloor$, corresponding to a median operation.

Two important special cases of MRL-NNs are obtained when $f$ is the identity, defining the MRL-NN of type I, and when $f$ is a nonlinearity of the logistic type, defining the MRL-NN of type II. In this way, an MLP is a special case of an MRL-NN of type II where $\lambda_n^{(l)} = 0 \forall n, l$, and an MRNN is a special case of an MRL-NN of type I where $\lambda_n^{(l)} = 1 \forall n, l$. Fig. 2 illustrates the structure of the $l$th layer of an MRNN [6].

[1] $\lfloor \cdot \rfloor$ denotes the usual truncation operation, so that $\lfloor \cdot + 0.5 \rfloor$ is the usual rounding operation.

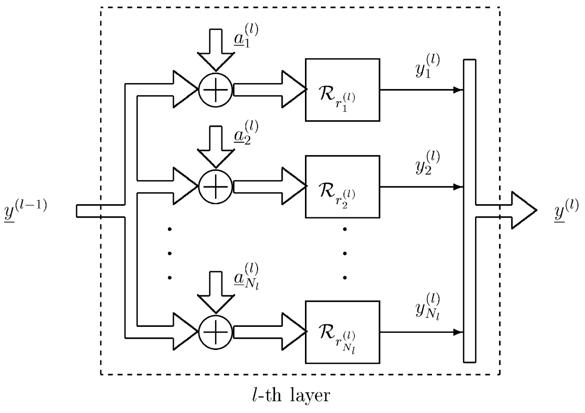


Fig. 2. Structure of the $l$th layer in an MRNN.

## 3. Geometrical insights

Structure (4) of every node in an MRL-NN is a compact representation of a set of hyperplanes. The normal vectors of those hyperplanes will depend on the mixing parameter $\lambda$ and the coefficients **b** of the linear FIR filter. If $\lambda = 1$, the hyperplanes are parallel to some subset of the canonical coordinate directions. For instance, consider a single-node MRL-NN in $\mathbb{R}^2$ with $r = 2$, i.e.,

$$y_1 = \lambda \min\{x_1 + a_1, x_2 + a_2\} + (1 - \lambda)(x_1 b_1 + x_2 b_2 + \tau_1). \tag{7}$$

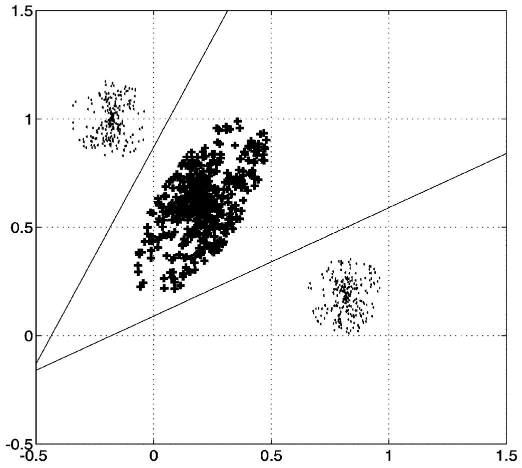
The boundary $y_1 = 0$ is defined by two lines obtained when either $\min\{x_1 + a_1, x_2 + a_2\} = x_1 + a_1$ or $\min\{x_1 + a_1, x_2 + a_2\} = x_2 + a_2$. The resulting lines are defined, respectively, by the equations

$$x_2 = \left[\frac{\lambda(b_1 - 1) - b_1}{(1 - \lambda)b_2}\right] x_1 - \left[\frac{\tau_1 + \lambda(a_1 - \tau_1)}{(1 - \lambda)b_2}\right],$$

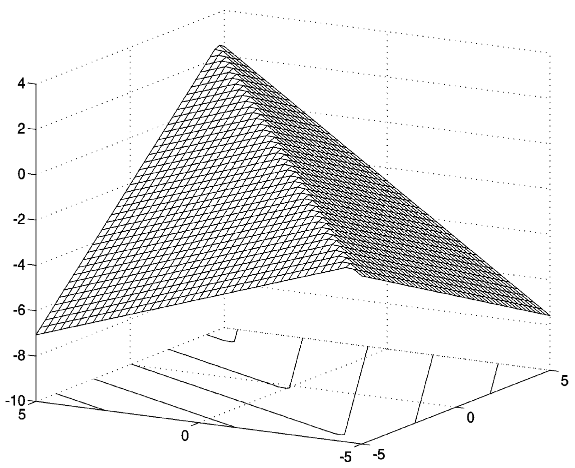
$$x_2 = \left[\frac{(\lambda - 1)b_1}{\lambda(1 - b_2) + b_2}\right] x_1 - \left[\frac{\tau_1 + \lambda(a_2 - \tau_1)}{\lambda(1 - b_2) + b_2}\right].$$

If these lines intercept each other, the intersection will occur along the line $x_2 = x_1 + a_1 - a_2$. It is not difficult to show that there will be no intersection if $b_1 + b_2 = \lambda/(\lambda - 1)$. Fig. 3(a) illustrates the use of the MRL-NN (7) to solve a two-class pattern recognition problem, where the corresponding six unknown parameters were estimated. Fig. 3(b) shows a plot of Eq. (7) as a function of $x_1$ and $x_2$. Similar classification could be obtained using a two-layer MLP with at least two hidden nodes, so that at least nine parameters would need to be estimated.

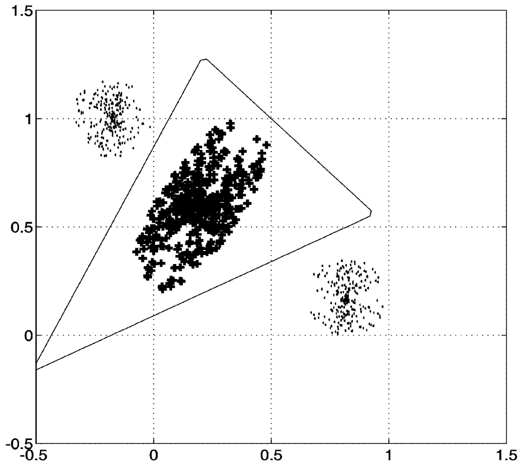
A possible way to improve results using a single node is obtained when we set $[\mathbf{x}, -\mathbf{x}]$ as the input signal. With this choice, we double the number of underlying



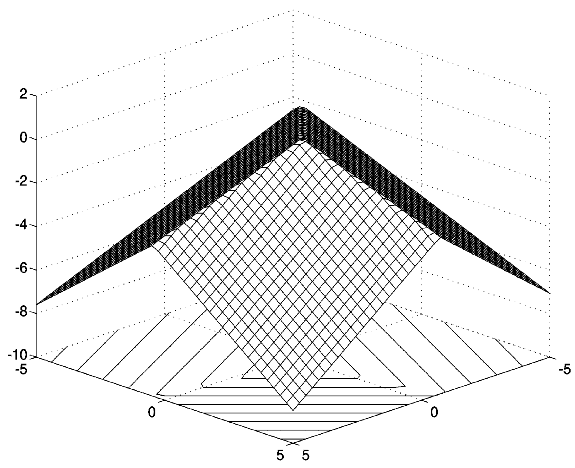
(a)



(b)



(c)



(d)

Fig. 3. Decision boundaries of MRL-NNs.

hyperplanes. For our example in $\mathbb{R}^2$, this means that we could easily obtain a closed boundary composed by four lines. Again, similar result could be obtained using a two-layer MLP with at least four hidden nodes. In terms of the number of parameters to be estimated, we would need 11 parameters in an MRL-NN and at least 17 parameters in an MLP.

Another solution to the classification problem is illustrated in Fig. 3(c), where now we use Eq. (7) to generate the two-layer MRL-NN

$$y_2 = \min\{y_1, b_3x_1 + b_4x_2 + \tau_2\}. \quad (8)$$

Observe that the resulting decision boundary is closed, and therefore provides robustness to reject spurious patterns [12]. Similarly, Fig. 3(d) shows a plot of Eq. (8) as a function of $x_1$ and $x_2$. For this MRL-NN we need to estimate nine parameters, whereas an MLP would need at least three hidden nodes to generate a closed region with three linear bounds, and the estimation of at least 13 parameters.

Thus, the MRL-NNs provide several improvements over MLPs. Not only the number of required nodes or parameters to be estimated in MRL-NNs can be smaller, but also sigmoid functions may not be necessary at all. Note also that, MRL-NNs provide improvements over MRNNs because the boundaries generated when $\lambda_n^{(l)} = 1 \forall n, l$ (i.e., when each node has no linear part) are located only in a finite number of directions. Therefore, the MRL-NN node has advantages over both the basic perceptron model as well as over the MRNN node. One drawback of MRL-NNs, however, is the computation of rank functions, but this is not a difficult task since in many pattern recognition applications the feature vectors to rank have a relatively small length and fast sorting algorithms are available.

## 4. The parity problem

The parity problem is a generalization of the XOR problem, i.e., given a binary vector $\mathbf{x}$ with $n$ components, the parity $P_n(\mathbf{x})$ is 1 if $\mathbf{x}$ contains an odd number of 1s and 0 otherwise. This problem is usually considered as a reference for checking new types of NNs or new training procedures.

Using an MLP trained by the back-propagation algorithm, the parity problem can be solved with at least $n$ hidden nodes [2], so that $n+1$ nodes are usually required and at least $(n+1)^2$ parameters need to be estimated. On the other hand, we can derive a closed-form solution to the parity problem using an MRL-NN. In fact, observe that for every binary vector $\mathbf{x}$ with $n$ components,

$$\sum_{r=1}^{n} \mathscr{R}_r(\mathbf{x}) = \mathbf{x}\cdot\mathbf{1}',$$

$$\sum_{r=1}^{n} (-1)^{r-1}\mathscr{R}_r(\mathbf{x}) = P_n(\mathbf{x}), \tag{9}$$

where $\mathbf{1} = (1, 1, \dots, 1)$. Thus, splitting the sums in Eq. (9) into even and odd values of $r$, yields

$$P_n(\mathbf{x}) = \mathbf{x}\cdot\mathbf{1}' - 2\sum_{r=1}^{\lfloor n/2 \rfloor} \mathscr{R}_{2r}(\mathbf{x}), \tag{10}$$

which clearly can be modeled by an MRL-NN of type I with only $\lfloor n/2 \rfloor + 2$ nodes, i.e., with about half of the number of nodes usually required by MLPs, and no more than $2n$ integer parameters. This result represents a considerable improvement over MLPs.

## 5. Adaptive design

Based on the LMS criterion and using ideas from the back-propagation algorithm, we propose a steepest descent method to optimally design general NNs and then apply it to MRL-NNs. The design goal is to achieve a set of parameters $\mathbf{w}_n^{(l)}$, $n = 1, 2, \dots, N_l$, $l = 1, 2, \dots, L$, such that some cost (or objective) function is minimized using a supervised procedure. Consider the training set

$$\{(\mathbf{x}(k), \mathbf{d}(k)),\ k = 0, 1, \dots, K-1\}, \tag{11}$$

where $\mathbf{d}(k)$ is the desired system output to the training sample $\mathbf{x}(k)$. From Eq. (11) we generate the training sequence[2]

$$(\mathbf{x}([k]_{\mathrm{mod}\ K}), \mathbf{d}([k]_{\mathrm{mod}\ K})), \quad k \in \mathbb{Z}, \tag{12}$$

by making a periodic extension of Eq. (11). Every period of Eq. (12) is usually called an *epoch*. A general supervised training algorithm is of the form

$$\mathbf{w}_n^{(l)}(i+1) = \mathbf{w}_n^{(l)}(i) + \mu_0 \mathbf{v}_n^{(l)}(i),\ \mu_0 > 0,$$

$$n = 1, 2, \dots, N_l;\ l = 1, 2, \dots, L, \tag{13}$$

where the positive constant $\mu_0$ controls the tradeoff between stability and speed of convergence, $\mathbf{v}_n^{(l)} = -\nabla J$, and $J$ is some cost function to be minimized. Let us define the error signal

$$\mathbf{e}(k) = (e_1(k), e_2(k), \dots, e_{N_L}(k)) = \mathbf{d}([k]_{\mathrm{mod}\ K}) - \mathbf{y}(k) \tag{14}$$

and the cost function

$$J(i) \equiv \frac{1}{M}\sum_{k=i-M+1}^{i} \xi(k),\ 1 \leqslant M \leqslant K, \tag{15}$$

where

$$\xi(k) \equiv \|\mathbf{e}(k)\|^2 = \sum_{n=1}^{N_L} e_n^2(k). \tag{16}$$

Based on the steepest descent algorithm, it follows from Eqs. (13) and (15) that

$$\mathbf{v}_n^{(l)}(i) = \frac{1}{M}\sum_{k=i-M+1}^{i} \mathbf{u}_n^{(l)}(k), \tag{17}$$

where

$$\mathbf{u}_n^{(l)}(k) = -\frac{\partial \xi(k)}{\partial \mathbf{w}_n^{(l)}}. \tag{18}$$

If we define the matrices $W^{(l)}$, $V^{(l)}$ and $U^{(l)}$ by

$$W^{(l)} \equiv \begin{pmatrix} \mathbf{w}_1^{(l)} \\ \mathbf{w}_2^{(l)} \\ \vdots \\ \mathbf{w}_{N_l}^{(l)} \end{pmatrix}, \quad V^{(l)} \equiv \begin{pmatrix} \mathbf{v}_1^{(l)} \\ \mathbf{v}_2^{(l)} \\ \vdots \\ \mathbf{v}_{N_l}^{(l)} \end{pmatrix}, \quad U^{(l)} \equiv \begin{pmatrix} \mathbf{u}_1^{(l)} \\ \mathbf{u}_2^{(l)} \\ \vdots \\ \mathbf{u}_{N_l}^{(l)} \end{pmatrix}, \tag{19}$$

[2] $[k]_{\mathrm{mod}\ K} \equiv k - K\lfloor k/K \rfloor$ denotes the index $k$ modulo $K$.

then algorithm Eq. (13) can be written as

$$W^{(l)}(i+1) = W^{(l)}(i) + \mu_0 V^{(l)}(i), \qquad l = 1, 2, \dots, L$$

$$V^{(l)}(i) = \frac{1}{M} \sum_{k=i-M+1}^{i} U^{(l)}(k). \tag{20}$$

In this way, using the chain rule to evaluate $U^{(l)}(k)$,

$$\frac{\partial \xi}{\partial \mathbf{w}_n^{(l)}} = \frac{\partial \xi}{\partial y_n^{(l)}} \frac{\partial y_n^{(l)}}{\partial z_n^{(l)}} \frac{\partial z_n^{(l)}}{\partial \mathbf{w}_n^{(l)}}. \tag{21}$$

Clearly, from Eq. (1) (see Fig. 1)

$$\frac{\partial y_n^{(l)}}{\partial z_n^{(l)}} = \dot{f}(z_n^{(l)}). \tag{22}$$

Let

$$\mathbf{e}^{(l)} \equiv -\frac{1}{2} \frac{\partial \xi}{\partial \mathbf{y}^{(l)}} \tag{23}$$

and

$$\boldsymbol{\gamma}_n^{(l)} \equiv \frac{\partial z_n^{(l)}}{\partial \mathbf{w}_n^{(l)}}. \tag{24}$$

Then, from Eqs. (22) to (24), Eq. (21) can be written as

$$\frac{\partial \xi}{\partial \mathbf{w}_n^{(l)}} = -2 e_n^{(l)} \dot{f}(z_n^{(l)})\, \boldsymbol{\gamma}_n^{(l)}. \tag{25}$$

Furthermore, define the local gradients by[3]

$$\boldsymbol{\delta}^{(l)} \equiv \mathbf{e}^{(l)} \odot \dot{F}(\mathbf{z}^{(l)}), \tag{26}$$

so that Eq. (25) can be written as

$$\frac{\partial \xi}{\partial \mathbf{w}_n^{(l)}} = -2 \delta_n^{(l)} \boldsymbol{\gamma}_n^{(l)}. \tag{27}$$

But from Eqs. (18) and (27) we conclude that

$$\mathbf{u}_n^{(l)} = 2 \delta_n^{(l)} \boldsymbol{\gamma}_n^{(l)}. \tag{28}$$

If we define the matrix $\Gamma^{(l)}$ by

$$\Gamma^{(l)} \equiv \begin{pmatrix} \boldsymbol{\gamma}_1^{(l)} \\ \boldsymbol{\gamma}_2^{(l)} \\ \vdots \\ \boldsymbol{\gamma}_{N_l}^{(l)} \end{pmatrix}, \tag{29}$$

then Eq. (28) can finally be written as

$$U^{(l)} = 2\, \mathrm{diag}(\boldsymbol{\delta}^{(l)}) \cdot \Gamma^{(l)}. \tag{30}$$

[3] We denote $\dot{F}(\mathbf{z}) \equiv (\dot{f}(z_1), \dot{f}(z_2), \dots, \dot{f}(z_n))$. The symbol '$\odot$' denotes an array (element-by-element) multiplication.

Notice that $\Gamma^{(l)}$ can be directly evaluated once we know the function $h(\cdot,\cdot)$. However, the local gradients $\boldsymbol{\delta}^{(l)}$ cannot be computed so simply. From Eq. (26), we observe that once we know $f(\cdot)$, then we can directly compute $\dot{F}(\cdot)$; but $\mathbf{e}^{(l)}$ is the problem. An efficient way to overcome this difficulty is to recursively compute $\mathbf{e}^{(l)}$ with a backward propagation.

Computing Eq. (23) for $l = L$, from Eqs. (16), (14) and (2) we obtain

$$\mathbf{e}^{(L)} = -\frac{1}{2} \frac{\partial \xi}{\partial \mathbf{y}^{(L)}} = \mathbf{e}. \tag{31}$$

For $l < L$, using the chain rule

$$\frac{\partial \xi}{\partial \mathbf{y}^{(l)}} = \sum_{n=1}^{N_{l+1}} \frac{\partial \xi}{\partial y_n^{(l+1)}} \frac{\partial y_n^{(l+1)}}{\partial \mathbf{y}^{(l)}},$$

so that from Eq. (23)

$$\mathbf{e}^{(l)} = \sum_{n=1}^{N_{l+1}} e_n^{(l+1)} \frac{\partial y_n^{(l+1)}}{\partial \mathbf{y}^{(l)}}. \tag{32}$$

Furthermore, from Eq. (1)

$$\frac{\partial y_n^{(l+1)}}{\partial \mathbf{y}^{(l)}} = \dot{f}(z_n^{(l+1)}) \frac{\partial z_n^{(l+1)}}{\partial \mathbf{y}^{(l)}}.$$

If we define

$$\boldsymbol{\theta}_n^{(l)} \equiv \frac{\partial z_n^{(l)}}{\partial \mathbf{y}^{(l-1)}}, \tag{33}$$

from Eqs. (32), (26) and (33) we have

$$\mathbf{e}^{(l)} = \sum_{n=1}^{N_{l+1}} \delta_n^{(l+1)} \boldsymbol{\theta}_n^{(l+1)}. \tag{34}$$

Therefore, if we define the matrix $\Theta^{(l)}$ by

$$\Theta^{(l)} \equiv \begin{pmatrix} \boldsymbol{\theta}_1^{(l)} \\ \boldsymbol{\theta}_2^{(l)} \\ \vdots \\ \boldsymbol{\theta}_{N_l}^{(l)} \end{pmatrix}, \tag{35}$$

then Eq. (34) can finally be written as

$$\mathbf{e}^{(l)} = \boldsymbol{\delta}^{(l+1)} \cdot \Theta^{(l+1)}. \tag{36}$$

The resulting training algorithm, called *the general averaged back-propagation algorithm*, is outlined in Table 1, where $\mu = 2\mu_0$. Observe that the central problem in using this general training algorithm is the evaluation of three derivatives: $\dot{f}$, $\boldsymbol{\theta}_n^{(l)} = \partial z_n^{(l)} / \partial \mathbf{y}^{(l-1)}$ and $\boldsymbol{\gamma}_n^{(l)} = \partial z_n^{(l)} / \partial \mathbf{w}_n^{(l)}$.

Table 1
General averaged back-propagation algorithm

(1) Set $i \leftarrow 0$. For $l = 1, 2, \ldots, L$, start the weights $W^{(l)}(i)$ with small uniformly distributed random variables.
(2) For $k = i - M + 1, i - M + 2, \ldots, i$ do

(a) Present the input $\mathbf{x}([k]_{\mathrm{mod}\ K})$ to the network, determine its output $\mathbf{y}(k)$ using Eqs. (1) and (2) (forward computation), and compute $\mathbf{e}(k)$ using Eq. (14).
(b) For $l = L, L-1, \ldots, 1$, evaluate the local gradients (backward computation)

$$\boldsymbol{\delta}^{(l)}(k) = \mathbf{e}^{(l)}(k) \odot \dot{F}(\mathbf{z}^{(l)}(k)),$$

where

$$\mathbf{e}^{(l)}(k) = \begin{cases} \mathbf{e}(k), & l = L, \\ \boldsymbol{\delta}^{(l+1)}(k) \cdot \Theta^{(l+1)}(k), & \text{otherwise} \end{cases}$$

and compute the gradient matrices

$$U^{(l)}(k) = \operatorname{diag}(\boldsymbol{\delta}^{(l)}(k)) \cdot \Gamma^{(l)}(k).$$

(3) Update the internal weights by

$$W^{(l)}(i+1) = W^{(l)}(i) + \mu V^{(l)}(i), l = 1, 2, \ldots, L$$

$$V^{(l)}(i) = \frac{1}{M} \sum_{k=i-M+1}^{i} U^{(l)}(k)$$

and set $i \leftarrow i + 1$.

(4) Repeat steps (2)–(3) until the weight values are stabilized and the cost function (15) is minimized, or until the maximum number of epochs to be processed is reached.

An important special case of the general averaged back-propagation algorithm is obtained when $N_L = 1$, corresponding to NN-filters. For this case, we have from Eq. (23)

$$\mathbf{e}^{(l)} = -\frac{1}{2} \frac{\partial \xi}{\partial \mathbf{y}^{(l)}} = -\frac{1}{2} \frac{\partial \xi}{\partial e} \frac{\partial e}{\partial y} \frac{\partial y}{\partial \mathbf{y}^{(l)}},$$

so that $\mathbf{e}^{(l)} = e\tilde{\mathbf{e}}^{(l)}$, where

$$\tilde{\mathbf{e}}^{(l)} \equiv \frac{\partial y}{\partial \mathbf{y}^{(l)}}. \tag{37}$$

In this way, defining

$$\tilde{\boldsymbol{\delta}}^{(l)} \equiv \tilde{\mathbf{e}}^{(l)} \odot \dot{F}(\mathbf{z}^{(l)}), \tag{38}$$

yields from Eq. (26) that $\boldsymbol{\delta}^{(l)} = e\tilde{\boldsymbol{\delta}}^{(l)}$. Similarly, defining

$$\tilde{U}^{(l)} \equiv 2 \operatorname{diag}(\tilde{\boldsymbol{\delta}}^{(l)}) \cdot \Gamma^{(l)}, \tag{39}$$

yields from Eq. (30) that $U^{(l)} = e\tilde{U}^{(l)}$. Therefore, recasting the above training algorithm with these tilded variables and setting $\mu = 2\mu_0$, simplifies the algorithm as outlined in Table 2.

Based on this framework, the design of MRL-NNs can easily be derived. From Eqs. (5) and (24),

$$\boldsymbol{\gamma}_n^{(l)} = \left( \frac{\partial z_n^{(l)}}{\partial \mathbf{a}_n^{(l)}}, \frac{\partial z_n^{(l)}}{\partial \rho_n^{(l)}}, \frac{\partial z_n^{(l)}}{\partial \mathbf{b}_n^{(l)}}, \frac{\partial z_n^{(l)}}{\partial \tau_n^{(l)}}, \frac{\partial z_n^{(l)}}{\partial \lambda_n^{(l)}} \right). \tag{40}$$

Using Eq. (4) of MRL-NNs, it follows from Eqs. (33) and (40) that

$$\boldsymbol{\theta}_n^{(l)} = \lambda_n^{(l)} \frac{\partial \alpha_n^{(l)}}{\partial \mathbf{y}^{(l-1)}} + (1 - \lambda_n^{(l)})\mathbf{b}_n^{(l)}, \tag{41}$$

$$\boldsymbol{\gamma}_n^{(l)} = \left( \lambda_n^{(l)} \frac{\partial \alpha_n^{(l)}}{\partial \mathbf{a}_n^{(l)}}, \lambda_n^{(l)} \frac{\partial \alpha_n^{(l)}}{\partial \rho_n^{(l)}}, (1 - \lambda_n^{(l)})\mathbf{y}^{(l-1)}, 1 - \lambda_n^{(l)}, \alpha_n^{(l)} - \beta_n^{(l)} \right). \tag{42}$$

The difficulty in evaluating Eqs. (41) and (42) is due to the nondifferentiability of rank functions, but we can circumvent this problem by using pulse functions as done in [5]. In this way,

$$\frac{\partial \alpha_n^{(l)}}{\partial \mathbf{y}^{(l-1)}} = \frac{\partial \alpha_n^{(l)}}{\partial \mathbf{a}_n^{(l)}} = \mathbf{c}_n^{(l)} \equiv \frac{Q(\alpha_n^{(l)}\mathbf{1} - \mathbf{y}^{(l-1)} - \mathbf{a}_n^{(l)})}{Q(\alpha_n^{(l)}\mathbf{1} - \mathbf{y}^{(l-1)} - \mathbf{a}_n^{(l)}) \cdot \mathbf{1}'} \tag{43}$$

and

$$\frac{\partial \alpha_n^{(l)}}{\partial \rho_n^{(l)}} = s_n^{(l)} \equiv 1 - \frac{1}{N_{l-1}} Q(\alpha_n^{(l)}\mathbf{1} - \mathbf{y}^{(l-1)} - \mathbf{a}_n^{(l)}) \cdot \mathbf{1}'. \tag{44}$$

In Eqs. (43) and (44), $Q(\mathbf{v}) \equiv (q(v_1), q(v_2), \ldots, q(v_n))$, where

$$q(v) \equiv \begin{cases} 1 & \text{if } v = 0, \\ 0 & \text{if } v \in \mathbb{R} \backslash \{0\}. \end{cases} \tag{45}$$

Table 2
General averaged back-propagation algorithm for NN-filters

```
(1) Set $i \leftarrow 0$. For $l = 1, 2, \ldots, L$, start the weights $W^{(l)}(i)$ with small uniformly distributed random variables.
(2) For $k = i - M + 1, i - M + 2, \ldots, i$ do
    (a) Present the input $\mathbf{x}([k]_{\text{mod } K})$ to the network, determine its output $y(k)$ using Eqs. (1) and (2) (forward computation), and compute $e(k)$ using Eq. (14).
    (b) For $l = L, L - 1, \ldots, 1$, evaluate the local gradients (backward computation)
        $\tilde{\boldsymbol{\delta}}^{(l)}(k) = \tilde{\mathbf{e}}^{(l)}(k) \odot \dot{F}(\mathbf{z}^{(l)}(k))$,
        where
        $\tilde{\mathbf{e}}^{(l)}(k) = \begin{cases} 1, & l = L, \\ \tilde{\boldsymbol{\delta}}^{(l+1)}(k) \cdot \Theta^{(l+1)}(k), & \text{otherwise} \end{cases}$
        and compute the gradient matrices
        $\tilde{U}^{(l)}(k) = \text{diag}(\tilde{\boldsymbol{\delta}}^{(l)}(k)) \cdot \Gamma^{(l)}(k)$.
(3) Update the internal weights by
    $W^{(l)}(i + 1) = W^{(l)}(i) + \mu V^{(l)}(i), \quad l = 1, 2, \ldots, L$
    $V^{(l)}(i) = \frac{1}{M} \sum_{k=i-M+1}^{i} e(k)\tilde{U}^{(l)}(k)$
    and set $i \leftarrow i + 1$.
(4) Repeat steps (2)–(3) until the weight values are stabilized and the cost function (15) is minimized, or until the maximum number of epochs to be processed is reached.
```

In order to avoid abrupt changes and achieve numerical robustness, we frequently replace the function $q(v)$ by smoothed impulses $q_\sigma(v)$, $\sigma \geqslant 0$, such as $\exp[-\frac{1}{2}(v/\sigma)^2]$ or $\text{sech}^2(v/\sigma)$ (see [5] for details).

The remaining unknown is $\dot{F}(\cdot)$, which depends on the type of MRL-NN in use. For the MRL-NN of type I, $F(\mathbf{z}^{(l)}) = \mathbf{z}^{(l)}$, so that $\dot{F}(\mathbf{z}^{(l)}) = \mathbf{1}$. For the MRL-NN of type II, we will use $f(z) = [1 + \exp(-\eta z)]^{-1}, \eta \geqslant 1$, whose derivative is $\dot{f}(z) = \eta f(z)[1 - f(z)]$, so that $\dot{F}(\mathbf{z}^{(l)}) = \eta \mathbf{y}^{(l)} \odot [\mathbf{1} - \mathbf{y}^{(l)}]$. In the next section, we will apply the general averaged back-propagation algorithm to design MRL-NNs and MLPs in problems of handwritten character recognition.

## 6. Applications in OCR

Using the design framework discussed in the previous section, we now present some representative experimental results from optical character recognition (OCR). Our approach is to perform a comparative analysis of MRL-NNs versus MLPs, illustrating some of the characteristics of both systems. We show that the MRL-NNs are a good alternative to MLPs, usually providing equal or better performance with smaller training times.

For our experiments, we have used a large database of handwritten characters provided by the National Institute of Standards and Technology (NIST) [13,14]. We selected a total of $K_T = 61{,}094$ samples of handwritten digits to form our data set. Those *digits* were generated and coded as follows [13]:

(1) A handwriting sample form (HSF) is filled out and scanned at a resolution of 300 dpi in binary format.
(2) The resulting image is processed and characters are extracted using connected components.
(3) Every valid character image is normalized in size to $20 \times 32$ pixels and then centered within a $32 \times 32$ image.
(4) A slant correction is applied to every normalized character image in an attempt to straighten its form.
(5) The resulting (binary) images are converted to vectors whose $32^2$ components are $+1$ for black pixels and $-1$ for white pixels.
(6) Principal component analysis (or Karhunen–Loève transform (KLT)) [15] is applied to generate feature vectors of dimension 64.

In our simulations, we normalized the feature vectors $\mathbf{x}(k)$, $k = 0, 1, \ldots, K_T - 1$, via the following transformation:

$$\mathbf{x}(k) \mapsto \frac{1}{2}\left(\frac{\mathbf{x}(k)}{\max_k \|\mathbf{x}(k)\|_\infty} + 1\right). \tag{46}$$

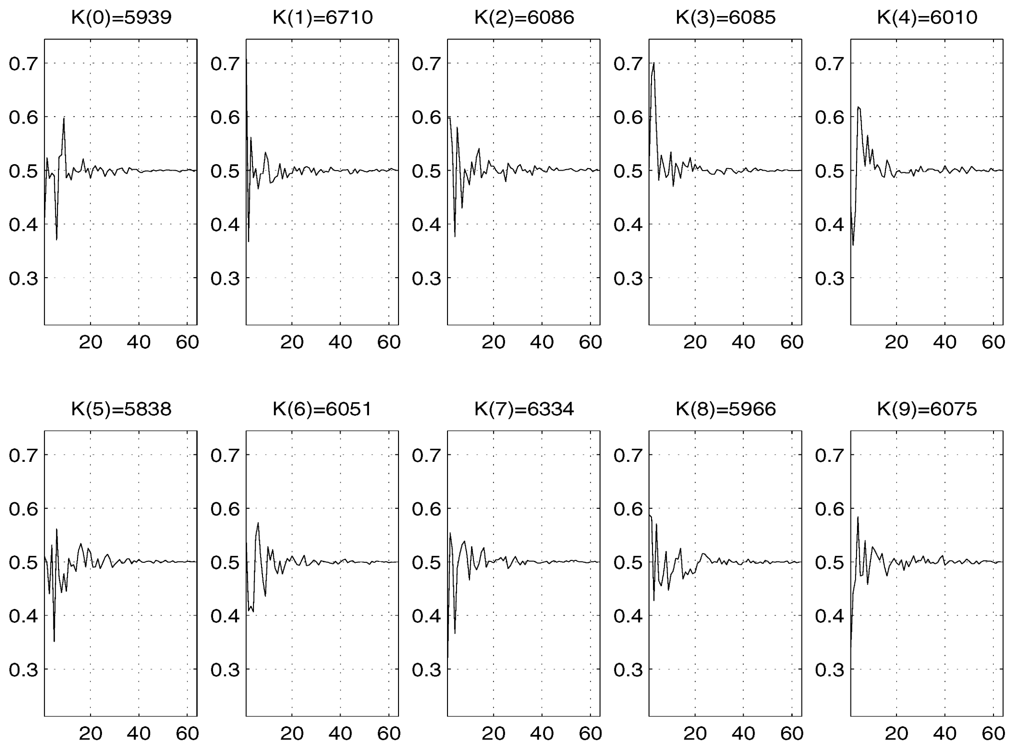


Fig. 4. Mean of the normalized KLT components.

For the selected data set, $\max_k \|\mathbf{x}(k)\|_\infty = 21.2$. Fig. 4 shows the mean of the resulting normalized vectors for every digit (0, 1, …, 9). In this figure, $K(i)$ indicates the number of elements in the class of digit $i$, so that $\sum_{i=0}^{9} K(i) = K_T$.

The data set was split such that 45,000 digits were used for training and the remaining 16,094 digits were used for testing (Test Set #1). The first 15,000 elements of the training set were used as a validation set during the training process, from which a convergence criterion was defined. The training sequence was ordered such that one instance of every digit is presented to the system in each iteration. Further, for the sake of a better understanding of the above coding process, we modified the HSF developed by the NIST and created a handwriting digits sample form (HDSF) to collect a second test set of digits. Then a group of 11 people were asked to fill out those forms, and a similar procedure was employed to generate another testing data (Test Set #2) composed of 132 examples of every digit. Results of this process are shown in Fig. 5. Note that the same KLT basis (provided by the NIST) was used to generate all feature vectors.

After making several tests, we designed a group of 12 experiments with three different network topologies: 64-N-10 MRL-NNs and MLPs with $N = 5, 10, 20$. This notation indicates a system with 64 inputs, $N$ hidden nodes, and 10 outputs. For simplicity, MRL5, MRL10 and MRL20 will denote MRL-NNs with 5, 10 and 20 hidden nodes (similarly for MLPs). Two different step sizes were tested: $\mu = 0.01, 0.1$. Every experiment was repeated 5 times (i.e., five runs) with different random initial conditions, and the best result is reported here. Among many possible ways to initialize the systems, and after performing various tests, we initialized the weights randomly in the ranges: $\mathbf{a}_n^{(l)}$: $[-0.1, 0.1]$, $r_n^{(l)}$: $[1, N_{l-1}]$, $b_n^{(l)}$: $[-1/\sqrt{N_{l-1}}, 1/\sqrt{N_{l-1}}]$, $\tau_n^{(l)}$: $[-0.1, 0.1]$, $\lambda_n^{(l)}$: $[0.4, 0.6]$. Further, in order to estimate gradients, we smoothed impulses with $q_\sigma(v) = \exp[-\frac{1}{2}(v/\sigma)^2]$, $\sigma = 0.05$. Due to the size of the training set, we have used the proposed training algorithm with $M = 1$ only. We have tested the case $M > 1$ with a small subset of the training set, but no significant improvements were observed. Both MRL-NNs and MLPs were defined using a sigmoid activation function with $\eta = 1$ (MRL-NNs of type II).

As usual, the desired system output $\mathbf{d} = (d_0, d_1, \ldots, d_9)$ was defined by

$$d_n = \begin{cases} 1 & \mathbf{x} \leftrightarrow \text{digit } n, \\ 0 & \text{otherwise.} \end{cases} \tag{47}$$

We say that an input digit $\mathbf{x}$ is misclassified if and only if the output $\mathbf{y}$ from the system has the property

$\text{argmax}\{\mathbf{y}\} \neq \text{argmax}\{\mathbf{d}\}$.

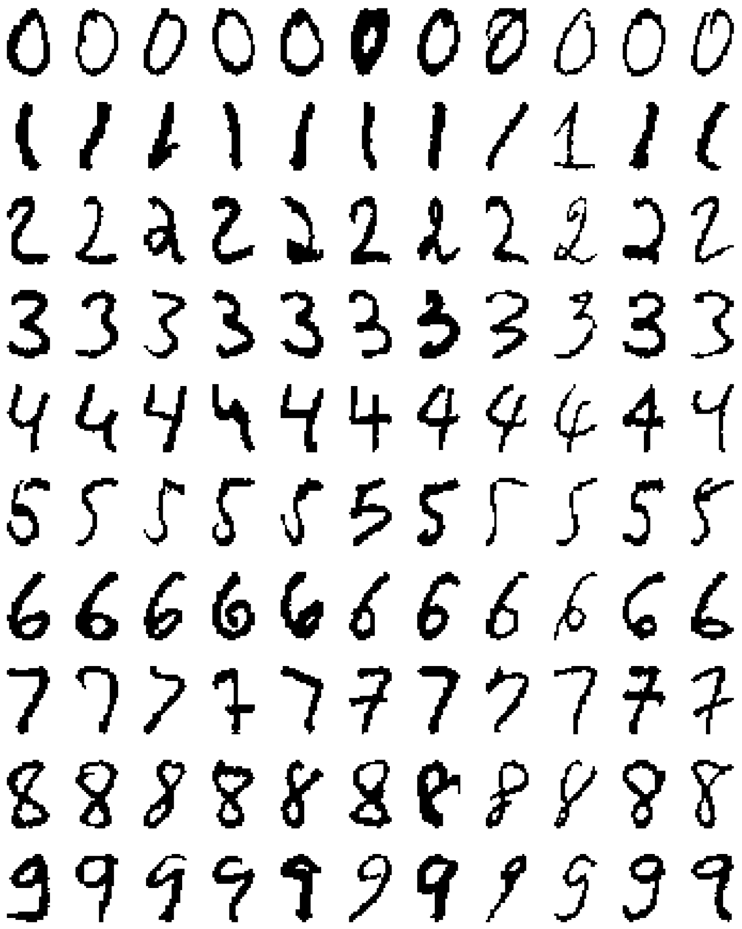


Fig. 5. Example of extracted digits from the completed HDSFs.

Further, for a given confidence threshold $\theta \in [0, 1]$, we say that an input digit $\mathbf{x}$ is rejected if and only if the output $\mathbf{y}$ from the system has the property

$$\operatorname{argmax}\{\mathbf{y}\} = \operatorname{argmax}\{\mathbf{d}\} \text{ AND } \max\{\mathbf{y}\} < \theta.$$

In this way, we define the rejection rate $R(\theta)$ and the error rate $E(\theta)$ (in a percentage basis) for a given set of digits by

$$R(\theta) = \frac{\#\text{ rejected digits}}{\text{total }\#\text{ digits}} \times 100, \tag{48}$$

$$E(\theta) = \frac{\#\text{ misclassified digits}}{\text{total }\#\text{ digits} - \#\text{ rejected digits}} \times 100. \tag{49}$$

Because $E$ and $R$ vary with $\theta$, we define their norms by

$$\|E(\theta)\| = \frac{1}{10}\sum_{i=0}^{9} E\left(\frac{i}{10}\right), \tag{50}$$

$$\|R(\theta)\| = \frac{1}{10}\sum_{i=0}^{9} R\left(\frac{i}{10}\right). \tag{51}$$

Using ideas from [16], two measurements were defined in the attempts to control the training process and to compare different systems. These measurements are the figure of merit (FM) and the generalization loss (GL), which are computed (for a given set) by the end of every epoch. The figure of merit for a given epoch $t$ is defined by the following convex combination of Eqs. (50) and (51):

$$\text{FM}(t) = \tfrac{1}{2}(\|E(\theta)\| + \|R(\theta)\|). \tag{52}$$

The training process tends to decrease the figure of merit, and good performance corresponds to small values of FM.

The generalization loss is used to stop the training process, and is defined as the relative increase of the figure of merit *in the validation set (vs)* over the minimum-so-far (in a percentage basis), i.e.,

$$\text{GL}(t) = \left(\frac{\text{FM}_{\text{vs}}(t)}{\min_{t' \leqslant t}\text{FM}_{\text{vs}}(t')} - 1\right) \times 100. \tag{53}$$

Table 3
Number of epochs of initial training and corresponding best figures of merit (out of each set of five runs) in the validation set when using $GL > 5\%$ or $t > 20$ as stopping criterion

| | $\mu = 0.01$ | | $\mu = 0.1$ | |
|---|---|---|---|---|
| | # Epochs | FM % | # Epochs | FM % |
| MRL5 | 11 | 23.6 | 4 | 17.2 |
| MLP5 | 20 | 23.3 | 19 | 14.7 |
| MRL10 | 17 | 18.2 | 9 | 15.2 |
| MLP10 | 20 | 16.8 | 20 | 12.9 |
| MRL20 | 10 | 16.3 | 4 | 13.4 |
| MLP20 | 20 | 14.7 | 11 | 13.2 |

A generalization loss larger than 5% or an epoch larger than 20 was our criterion to stop training during the initial stage of five runs per experiment.

Using our proposed training algorithm with all the above considerations, we observed the results presented in Table 3. Notice that, either for a step size $\mu = 0.01$ or 0.1, the MRL-NNs required a smaller number of iterations than MLPs, and provided similar performances (FMs) in the validation set. The reported values of FM are the minimum-so-far up to the epoch when the training stopped. Also, the internal parameters of the systems to be used after training correspond to those minima. Computing the figures of merit of MLPs with equal number of iterations of MRL-NNs, we usually observed better performances with MRL-NNs.

Next, we continued the training processes of the experiments indicated in Table 3 for 100 epochs (independently of GL) and analyzed the classification results obtained with the systems of minimum figures of merit. We observed that the training processes of MRL-NNs usually converge faster than MLPs, but at a price of an oscillating behavior of the figures of merit. MLPs, on the other hand, have a smooth convergence behavior, but their training processes usually take a longer time to reach optimal solutions. Detailed plots of FM and GL versus number of epochs for both MRL-NNs and MLPs, and of error and rejection rates versus confidence threshold for training set and test set (Test Set #1) are not included here due to space limitation, but can be found in [4].

Table 4 summarizes our best simulation results in terms of training, testing and overall performance. The rows with epochs indicate the amount of time necessary to reach the minimum FMs. Again, we usually observed

Table 4
Figure of merit/error rate/rejection rate (%) corresponding to the optimal set of weights of best MRL-NN versus best MLP for $\mu = 0.01$, 0.1

| | FM/$\|E(\theta)\|$/$\|R(\theta)\|$ | | | |
|---|---|---|---|---|
| | $\mu = 0.01$ | | $\mu = 0.1$ | |
| | MRL5 | MLP5 | MRL5 | MLP5 |
| Training | 10.7/11.5/10.0 | 12.5/11.8/13.1 | 11.8/13.2/10.5 | 9.9/8.8/11.1 |
| Test1 | 14.7/16.8/12.5 | 16.8/12.7/20.9 | 18.7/22.4/15.0 | 18.4/19.9/16.9 |
| Test2 | 25.7/25.2/26.2 | 29.7/22.0/37.3 | 21.0/20.3/21.7 | 23.6/17.4/30.0 |
| Overall | 12.0/13.1/11.0 | 13.9/12.2/15.6 | 13.7/15.6/11.9 | 12.3/11.6/13.0 |
| Epoch | 24 | 100 | 3 | 10 |
| | MRL10 | MLP10 | MRL10 | MLP10 |
| Training | 13.8/13.0/14.5 | 8.2/6.5/9.9 | **7.4**/6.9/7.8 | **7.4**/7.0/7.7 |
| Test1 | 13.0/13.6/12.5 | 13.6/12.5/14.7 | **11.0**/13.1/8.9 | **11.1**/10.9/11.4 |
| Test2 | 23.0/23.8/22.0 | 20.4/13.8/27.0 | **16.5**/13.7/19.2 | **19.1**/18.6/19.6 |
| Overall | 9.2/8.8/9.6 | 9.8/8.2/11.5 | **8.5**/8.7/8.3 | **8.6**/8.2/8.9 |
| Epoch | 99 | 96 | **62** | **96** |
| | MRL20 | MLP20 | MRL20 | MLP20 |
| Training | 7.6/8.4/6.9 | 7.5/6.9/8.1 | 8.4/7.8/9.0 | **7.5**/6.8/8.2 |
| Test1 | 12.4/15.0/9.7 | 13.4/12.3/14.5 | 17.4/24.6/10.2 | **11.8**/12.8/10.9 |
| Test2 | 20.1/23.1/17.1 | 20.8/13.3/28.3 | 19.1/19.5/18.8 | **20.5**/20.6/20.4 |
| Overall | 9.1/10.4/7.8 | 9.3/7.3/11.4 | 10.9/12.3/9.6 | **8.9**/8.7/9.1 |
| Epoch | 100 | 100 | 9 | **88** |

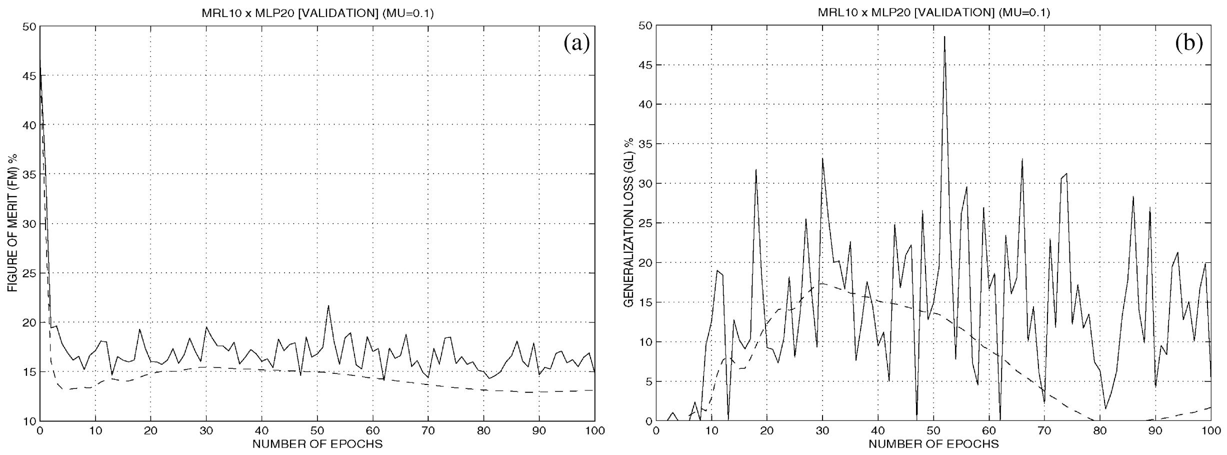


Fig. 6. MRL10 (solid line) versus MLP20 (dashed line): figure of merit and generalization loss.

Table 5
Confusion matrices for the training set generated with the optimal set of weights of best MRL10 versus best MLP20 when $\mu = 0.1$ and $\theta = 0.5$

| Actual digit | MRL10 classification output | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | Rejected |
| 0 | 4487 | 0 | 0 | 0 | 1 | 1 | 8 | 0 | 0 | 0 | 3 |
| 1 | 0 | 3751 | 13 | 48 | 14 | 32 | 60 | 31 | 61 | 0 | 490 |
| 2 | 274 | 4 | 3455 | 108 | 9 | 1 | 155 | 2 | 49 | 0 | 443 |
| 3 | 16 | 0 | 17 | 4323 | 3 | 13 | 6 | 2 | 8 | 5 | 107 |
| 4 | 47 | 6 | 8 | 8 | 3927 | 0 | 109 | 0 | 44 | 14 | 337 |
| 5 | 113 | 0 | 1 | 150 | 6 | 3425 | 255 | 0 | 3 | 3 | 544 |
| 6 | 171 | 0 | 0 | 0 | 0 | 0 | 4321 | 0 | 0 | 0 | 8 |
| 7 | 73 | 2 | 1 | 2 | 9 | 0 | 0 | 4093 | 8 | 63 | 249 |
| 8 | 81 | 58 | 9 | 268 | 12 | 71 | 28 | 15 | 3399 | 16 | 543 |
| 9 | 39 | 0 | 0 | 20 | 14 | 11 | 4 | 104 | 20 | 3531 | 757 |

| Actual digit | MLP20 classification output | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | Rejected |
| 0 | 4494 | 0 | 0 | 0 | 3 | 0 | 0 | 0 | 0 | 0 | 3 |
| 1 | 5 | 3709 | 6 | 93 | 4 | 40 | 11 | 51 | 55 | 0 | 526 |
| 2 | 268 | 13 | 3096 | 188 | 13 | 8 | 77 | 17 | 20 | 0 | 800 |
| 3 | 11 | 0 | 1 | 4366 | 1 | 29 | 1 | 5 | 3 | 8 | 75 |
| 4 | 106 | 9 | 5 | 22 | 4065 | 8 | 14 | 2 | 13 | 9 | 247 |
| 5 | 119 | 0 | 0 | 196 | 5 | 3923 | 44 | 2 | 1 | 6 | 204 |
| 6 | 269 | 0 | 0 | 0 | 14 | 5 | 4020 | 0 | 0 | 0 | 192 |
| 7 | 26 | 1 | 0 | 8 | 6 | 2 | 0 | 4181 | 3 | 26 | 247 |
| 8 | 199 | 64 | 6 | 273 | 11 | 101 | 10 | 13 | 3122 | 37 | 664 |
| 9 | 134 | 1 | 0 | 18 | 11 | 4 | 0 | 71 | 8 | 3823 | 430 |

similar performances of MRL-NNs and MLPs, but with smaller number of iterations for MRL-NNs. The overall best result was FM = 8.5%, obtained with a 64-10-10 MRL-NN. Similar results were obtained with a 64-10-10 MLP (FM = 8.6%) and a 64-20-10 MLP (FM = 8.9%), but with a larger number of iterations.

Finally, we compare MRL10 versus MLP20 (for $\mu = 0.1$), because the two systems have about the same

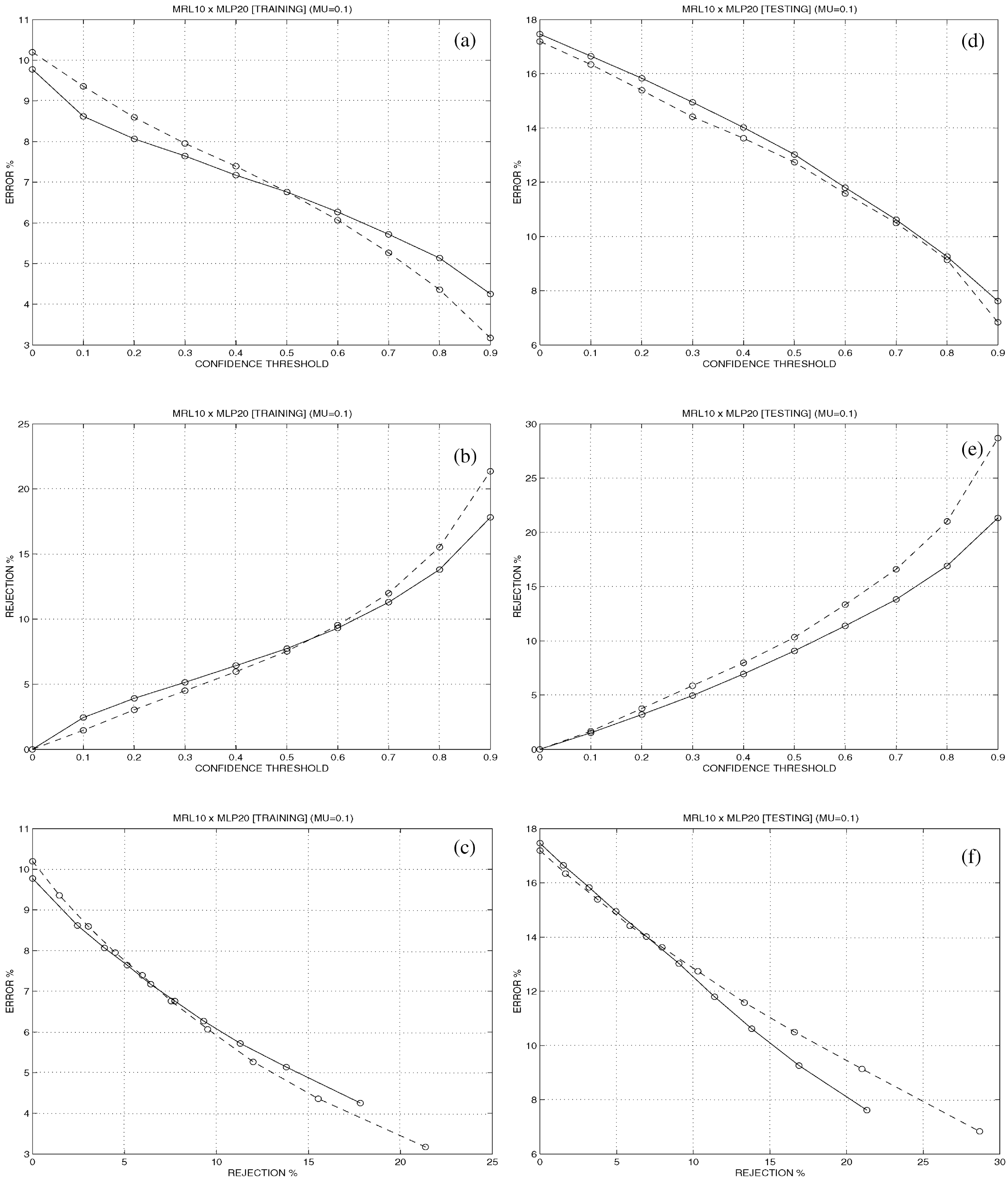


Fig. 7. MRL10 (solid line) versus MLP20 (dashed line): error and rejection rates.

Table 6
Confusion matrices for the test set generated with the optimal set of weights of best MRL10 versus best MLP20 when $\mu = 0.1$ and $\theta = 0.5$

| Actual digit | MRL10 classification output | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | Rejected |
| 0 | 1141 | 17 | 6 | 44 | 38 | 0 | 46 | 3 | 5 | 2 | 137 |
| 1 | 0 | 2171 | 2 | 1 | 0 | 0 | 0 | 22 | 6 | 0 | 8 |
| 2 | 0 | 23 | 1178 | 42 | 9 | 0 | 0 | 38 | 70 | 0 | 226 |
| 3 | 0 | 4 | 13 | 1289 | 0 | 0 | 0 | 116 | 36 | 26 | 101 |
| 4 | 0 | 37 | 0 | 0 | 869 | 0 | 0 | 5 | 222 | 42 | 335 |
| 5 | 1 | 14 | 6 | 19 | 0 | 1063 | 2 | 9 | 50 | 7 | 167 |
| 6 | 5 | 15 | 24 | 11 | 3 | 0 | 1360 | 0 | 0 | 0 | 133 |
| 7 | 0 | 10 | 3 | 0 | 0 | 0 | 0 | 1731 | 59 | 5 | 26 |
| 8 | 0 | 127 | 0 | 2 | 0 | 0 | 0 | 21 | 1276 | 0 | 40 |
| 9 | 0 | 10 | 0 | 0 | 1 | 0 | 0 | 419 | 208 | 651 | 286 |

| Actual digit | MLP20 classification output | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | Rejected |
| 0 | 1125 | 23 | 3 | 27 | 48 | 1 | 20 | 2 | 0 | 7 | 183 |
| 1 | 0 | 2178 | 0 | 2 | 0 | 0 | 0 | 11 | 12 | 0 | 7 |
| 2 | 0 | 18 | 992 | 112 | 8 | 0 | 0 | 60 | 19 | 0 | 377 |
| 3 | 0 | 9 | 5 | 1315 | 0 | 0 | 0 | 148 | 22 | 20 | 66 |
| 4 | 0 | 31 | 0 | 0 | 923 | 3 | 0 | 30 | 45 | 45 | 433 |
| 5 | 4 | 12 | 2 | 18 | 1 | 1119 | 0 | 37 | 24 | 8 | 113 |
| 6 | 12 | 25 | 13 | 0 | 14 | 3 | 1337 | 0 | 2 | 0 | 145 |
| 7 | 0 | 17 | 0 | 0 | 1 | 1 | 0 | 1637 | 150 | 2 | 26 |
| 8 | 0 | 139 | 0 | 2 | 0 | 0 | 0 | 10 | 1262 | 3 | 50 |
| 9 | 0 | 20 | 0 | 0 | 1 | 3 | 0 | 333 | 250 | 708 | 260 |

number of internal weights. Fig. 6 shows the plots of FM and GL versus number of epochs for both systems, and Fig. 7 illustrates the error and rejection rates versus confidence threshold for training and test sets. Tables 5 and 6 show the confusion matrices of training and test sets of best MRL10 versus best MLP20 for $\mu = 0.1$ and $\theta = 0.5$ (confidence threshold).

Comparing our best MRL-NN (a 64-10-10 NN) with the best NIST MLP [14] (a 128-128-10 NN), we have that with no rejection (i.e., $\theta = 0$) their system has an error rate near to 4% on some data set while our system has a training error rate of about 10%. However, both the network topologies and the training algorithms are different.

We used MLPs as our benchmark because they represent a subset of MRL-NNs (with no morphological component). An exhaustive comparison of different neural network architectures, such as radial basis function and associative memory-type networks, in problems of handwritten character recognition is beyond the scope of this paper. The reader should keep in mind that other alternative architectures and/or different choices of feature vectors could generate results comparable to or better than those within this paper.

## 7. Conclusions

The general class of MRL-NNs and its training algorithm were presented in this paper. The fundamental processing unit of this class of systems is the MRL-filter, i.e., a linear combination between a morphological/rank filter and a linear FIR filter. Every node in an MRL-NN is a shifted MRL-filter followed by some activation function. This node structure is viewed as an extension of the basic perceptron model.

The usefulness of MRL-NNs is illustrated by showing that the parity problem can be solved in closed form with about half of the number of nodes usually required by MLPs and a smaller computational complexity. Examples from simple pattern classification problems are also included to provide geometrical insights for the proposed NN architecture. The MRL-NNs have the unifying property that the characteristics of both MLPs and MRNNs are observed in the same system.

An important general result was the formulation of a systematic design framework based on the back-propagation algorithm, which is fully defined by evaluating only three derivatives. This framework was used to derive the training algorithm of MRL-NNs, but it can also be directly applied to design other types of NNs.

Finally, we applied the proposed design methodology to problems of optical character recognition and provided extensive experimental evidences showing not only that the MRL-NNs can generate similar or better results when compared with the classical MLPs, but they also usually require smaller processing times for training. We used a large database from the National Institute of Standards and Technology and, after exhaustive analysis and simulations, the best overall result was obtained with our proposed system. The MRL-NNs define a broad and interesting class of nonlinear systems with many promising applications in signal/image processing and pattern recognition.

## References

[1] S. Marcos, O. Macchi, C. Vignat, G. Dreyfus, L. Personnaz, P. Roussel-Ragot, A unified framework for gradient algorithms used for filter adaptation and neural network training, Int. Circuit Theory Appl. 20 (1992) 159–200.

[2] D.E. Rumelhart, G.E. Hinton, R.J. Willians, Learning internal representations by error propagation, in: D.E. Rumelhart, J.L. McClelland (Eds.), Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1, MIT Press, Cambridge, MA, 1986, pp. 318–362 (Chapter 8).

[3] G.M. Shepherd, R.K. Brayton, Logic operations are properties of computer-simulated interactions between excitable dendritic spines, Neuroscience 21 (1) (1987) 151–165.

[4] L.F.C. Pessoa, Nonlinear systems and neural networks with hybrid morphological/rank/linear nodes: optimal design and applications to image processing and pattern recognition, Ph.D. Thesis, Georgia Institute of Technology, Atlanta, GA, 1997.

[5] L.F.C. Pessoa, P. Maragos, MRL-Filters: a general class of nonlinear systems and their optimal design for image processing, IEEE Trans. Image Process. 7 (7) (1998) 966–978.

[6] L.F.C. Pessoa, P. Maragos, Morphological/rank neural networks and their adaptive optimal design for image processing, Proc, IEEE Int. Conf. Acoust. Speech Signal Process. 6 (1996) 3399–3402.

[7] P. Yang, P. Maragos, Min–max classifiers: learnability, design and application, Pattern Recognition 28 (6) (1995) 879–899.

[8] W. Pedrycz, Neuralcomputations in relational systems, IEEE Trans. Pattern Anal. Mach. Intell. 13 (3) (1991) 289–297.

[9] X. Zhang, C.-C. Hang, S. Tan, P.-Z. Wang, The min–max function differentiation and training of fuzzy neural networks, IEEE Trans. Neural Networks 7 (5) (1996) 1139–1150.

[10] P. Salembier, Adaptive rank order based filters, Signal Processing 27 (1992) 1–25.

[11] P. Salembier, Structuring element adaptation for morphological filters, J. Visual Commun. Image Representation 3 (1992) 115–136.

[12] G.C. Vasconcelos, M.C. Fairhurst, D.L. Bisset, Investigating feedforward neural networks with respect to the rejection of spurious patterns, Pattern Recognition Lett. 16 (2) (1995) 207–212.

[13] M.D. Garris, J.L. Blue, G.T. Candela, D.L. Dimmick, J. Geist, P.J. Grother, S.A. Janet, C.L. Wilson, NIST form-based handprint recognition system, Technical Report NISTIR 5469, National Institute of Standards and Technology, 1994.

[14] M.D. Garris, J.L. Blue, G.T. Candela, P.J. Grother, S.A. Janet, C.L. Wilson, NIST form-based handprint recognition system (release 2.0), Technical Report NISTIR 5959, National Institute of Standards and Technology, 1997.

[15] I.T. Jolliffe, Principal Component Analysis, Springer, Berlin, 1986.

[16] L. Prechlet, PROBEN1 — a set of neural network benchmark problems and benchmarking rules, Technical Report 21/94, Universität Karlsruhe, 1994.

**About the Author**—LÚCIO F.C. PESSOA received the B.S.E.E. and the M.S.E.E. degrees from the Federal University of Pernambuco (UFPE), Recife, Brazil, in 1990 and 1992, respectively, and the Ph.D. degree from the Georgia Institute of Technology (Georgia Tech), Atlanta, in 1997. From 1995 to 1997, he was a Graduate Research Assistant in the Center for Signal and Image Processing (formerly known as DSP Laboratory) of the School of Electrical and Computer Engineering, Georgia Tech. In March 1998, he joined the Broadband Division of Motorola Semiconductor Products Sector, Austin, TX, where he is currently a member of the Group for Advanced Architectures and Algorithms, working with DSP research and development for digital subscriber line (xDSL) systems. His Ph.D. research project was in the design of optimal morphological systems with applications to image processing and neural networks. His main research interests include image processing, computer vision, pattern recognition, and digital communications. Dr. Pessoa was honored with several research fellowships during his studies at UFPE, and in 1990 he received the Brazilian IEEE/CEMIG award for the paper "Pattern Generation: An Alternative to Steady-State Security Evaluation in Power Systems". He is a member of IEEE.

**About the Author**—PETROS MARAGOS received the Diploma degree in electrical engineering from the National Technical University of Athens, Greece, in 1980, and the M.S.E.E. and Ph.D. degrees from the Georgia Institute of Technology, Atlanta, in 1982 and 1985. In 1985, he joined the faculty of the Division of Applied Sciences at Harvard University, Cambridge, MA, where he worked as Assistant (1985–1989) and Associate Professor (1989–1993) of electrical engineering, affiliated with the interdisciplinary Harvard Robotics Laboratory. He has also been a consultant to several industry research groups including Xerox's research on document image analysis. In 1993, he joined the faculty of the Georgia Institute of Technology. During parts of 1996–1998, he has also worked as a Senior

Researcher at the Institute for Language and Speech Processing in Athens. In 1998, he joined the faculty of the National Technical University of Athens where he is a Professor of electrical and computer engineering. Dr. Maragos' research work has received several awards, including a 1987 U.S. National Science Foundation Presidential Young Investigator Award; the 1988 IEEE Signal Processing Society's Paper Award for the paper "Morphological Filters"; the 1994 IEEE Signal Processing Society's Senior Award and the 1995 IEEE Baker Award for the paper "Energy Separation in Signal Modulations with Application to Speech Analysis" (co-recipient); and the 1996 Pattern Recognition Society's Honorable Mention Award for the paper "Min-Max Classifiers" (co-recipient). He has also served as Associate Editor for the IEEE Transactions on Signal Processing and Guest Editor for the IEEE Transactions on Image Processing; General Chairman for the 1992 SPIE Conference on Visual Communications and Image Processing, and Co-chairman for the 1996 International Symposium on Mathematical Morphology. He has been a member of two IEEE DSP Committees and President of the International Society for Mathematical Morphology. He is a Fellow of IEEE.