Applied Mathematical Sciences, Vol. 2, 2008, no. 26, 1253 - 1282

$\Theta(1)$ Time Neural Network Minimum Distance Classifier and its Application to Optical Character Recognition Problem

A. Namir

Département de Mathématiques et Informatique
Faculté des sciences Ben M'sik Université Hassan II, Mohammedia
B.P 7955, Sidi Othman, Casablanca, Marocco
a.Namir@yahoo.fr

M. Mestari

E.N.S.E.T Mohammedia Avenue Hssan II Mohammedia Morocco

K. Akodadi and A. Badi

Département de Mathématiques et Informatique
Faculté des sciences Ben M'sik Université Hassan II, Mohammedia
B.P 7955, Sidi Othman, Casablanca, Marocco

Abstract

We propose a special neural network model, NNC, (Neural Network Classifier), which with a classification problem of l classes $C_1, C_2, ..., C_l$, classifies an unknown vector to one class using a minimum distance classification technique. The NNC consists of three kinds of neurons, linear, quasi-linear and threshold-logic neurons, distributed over 12 layers. Thus, its processing time is constant and 12 times that of a single neuron. NNC architecture is regular and simple, it can be easily implemented using VLSI technology. High performance of NNC is demonstrated by applying it to an optical character recognition problem.

Keywords: neural network classifier (NNC); minimum distance classification (MDC); optical character recognition (OCR)

1 Introduction

Minimum distance classification (MDC) is a simple yet powerful technique

widely used in statistical recognition, clustering and many other applications (such as pattern recognition and image processing (PRIP) applications) Bahl et al. [1], David and Andrew, 1991 [3], Kung and Taur [10], Li and Dong [11], Philip [21], Sethi and Sarvarayudu [22], Shore and Gray [23]. During the past decades, considerable efforts have been devoted to developing special computer architectures for PRIP. Recent advances in VLSI microelectronic technology have triggered the idea of implementing PRIP algorithms directly in specialized hardware chips. In this paper, our objective is to design a special-purpose architecture of high processing speed which can be used for implementing (MDC) technique. NNC network is best suited to achieve this goal and is hardware implementable with very simple circuit elements, such as resistors and linear operational amplifiers Mestari [16]. Unlike the digital parallel algorithms the proposed neural network solves the classification problem without using floating-point arithmetics, the CPU (central processing unit), the memory chips, and the synchronization clock.

NNC performs classification using minimum distance technique and has a feedforward structure with three kinds of neurons, linear, quasi-linear and threshold-logic neurons, displayed over 12 layers. Its overall processing time is constant and 12 times that of a single neuron. This is in contrast with conventional systolic array implementation, where processing time increases with data size.

Three types of neurons are employed in NNC. All of them have been used in constructing various kinds of neural networks Cichoki and Unbehauen [2], Fukushima [5], Hopfield [7], Kohonen [9], Mestari and Namir [13], Mestari et al. [14], Mestari and Namir [15], Mestari [16], Mestari and Namir [17], Mestari and Namir [18], Mestari [19], Mestari [20]. Among all the neurons proposed in the literature, they are probably the easiest to implement in hardware.

The approach to NNC construction herein proposed differs from that conventially taken in the neural network field, where the solution to any given problem necessitates taking a general purpose network and applying it by learning Kung and Taur [10], William et al. [24], Frey and State [4]. In this approach, our primary concern lies in how to organize neurons into a network which can solve a specific problem with an emphasis on fully utilizing the massive parallelism property offered by neural networks.

The rest of the paper is organized as follows: Section 2 describes the construction of NNC. Section 3 gives an example of applying NNC to optical character recognition (OCR) problem. Section 4 contains the conclusions.

2 Construction Of NNC

In this section we show how NNC is constructed. But first, we introduce a special network AMAXNET (Adjustable MAXNET) which is essential for

construction. This network is described in subsection 2.2 after description of the neurons employed in NNC and AMAXNET networks.

2.1 Artificial Neurons Used

NNC and AMAXNET employ three kinds of neuron, all of which are commonly used in neural network applications Fukushima [5], Kohonen [9], Lippman [12], Mestari and Namir [13], Mestari et al. [14], Mestari and Namir [15], Mestari [16], Mestari and Namir [17], Mestari and Namir [18], Mestari [19], Mestari [20]. The only difference amongst these three kinds of neurons is in their activation function, one employees the linear activation function, one the quasi-linear activation function, and the other the threshold-logic activation function. They are schematically illustrated in Fig. 1 (b).

These three kinds of neurons sum the n weighted inputs and pass the result through a nonlinearity according to the equation

$$y = \Phi\left(\sum_{i=1}^{n} \omega_i x_i \; - \; \theta\right) \tag{1}$$

Where Φ is a limiting or nonlinear transfer characteristic, called an activation function, θ ($\theta \in \mathbf{R}$) is the external threshold, also called an offset or bias, ω_i are the synaptic weights or strengths, x_i are the inputs ($i = 1, 2, ..., n$), n is the number of inputs and y represents the output (Cf. Fig. 1 (a)).

The linear activation function is defined as

$$\Phi_L(x) \stackrel{\text{def}}{=} x \tag{2}$$

the quasi-linear activation function is defined as

$$\Phi_Q(x) \stackrel{\text{def}}{=} \begin{cases} x & \text{if } x \geq 0 \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

and the threshold-logic activation function is defined as

$$\Phi_T(x) \stackrel{\text{def}}{=} \begin{cases} 1 & \text{if } x \geq 0 \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

where $x = \sum_{i=1}^{n} w_i x_i - \theta$ and n is the number of inputs.

All three types of neuron have already been implemented in the past using analogue electronic circuits. A typical implementation of the linear neuron uses a linear operational amplifier Hopfield and Tank [6]. The quasi-linear neuron may be implemented using an op amp/diode active clamp circuit Hopfield and Tank [6] or using a nonlinear resistor Kennedy and Chua [8]. For the threshold-logic neuron, many different schemes for implementation have been reported.

2.2 Construction of AMAXNET

A neural network, AMAXNET, which, with an array of real numbers as input, outputs the k^{th} largest element of the array is proposed. Overall processing time is constant, and does not depend upon the size of the input array, being just six times the processing time for a single neuron. First, however, two definitions which are essential for construction of AMAXNET are introduced.

Definition 2.1 *The order in the input array* $X = (x_1, x_2, ..., x_n)$ *of any element* x_i $(1 \leq i \leq n)$ *is defined as being the number of elements of the input array less than* x_i.

Definition 2.2 *Let* x_i *and* x_q *be two elements of the same value. If* $(i \leq q)$, *then* x_i *is considered smaller than* x_q, *otherwise* x_i *is considered larger than* x_q.

AMAXNET principle may be summarized thus: Overall, AMAXNET is composed of n order networks, ON_j $(1 \leq j \leq n)$, and a selection network, SN. Also, AMAXNET is equipped with a control input called an adjustment input. This adjustment input allows choice of the order of the element to be transferred to output.

The function of the order networks ON_j $(1 \leq j \leq n)$ is to compute the order of each element x_j in the input array. The selection network's function, however, is to select and transfer to output the element of the input array corresponding to the order desired or chosen via the adjustment input.

2.2.1 Order Network

Let x_i and x_q be two elements of the input array X. The comparison function of x_i and x_q is defined as follows:
for $i \in \{1, ..., q-1\}$,

$$\begin{aligned} comp(x_i, x_q) &\stackrel{\text{def}}{=} \begin{cases} 1 & \text{if } x_q \geq x_i \\ 0 & \text{otherwise} \end{cases} \\ &= \Phi_T(x_q - x_i) \end{aligned} \tag{5}$$

and for $i \in \{q+1, ..., n\}$,

$$\begin{aligned} comp(x_i, x_q) &\stackrel{\text{def}}{=} \begin{cases} 1 & \text{if } x_q > x_i \\ 0 & \text{otherwise} \end{cases} \\ &= \Phi_T(-\Phi_T(x_i - x_q)) \end{aligned} \tag{6}$$

The order of any element x_q in the input array X may be given by the following order function:

$$\begin{aligned} ord(x_q, X) &\stackrel{\text{def}}{=} \sum_{i<q} comp(x_i, x_q) + \sum_{i>q} comp(x_i, x_q) + 1 \\ &= \sum_{i<q} \Phi_T(x_q - x_i) + \sum_{i>q} \Phi_T(-\Phi_T(x_i - x_q)) + 1 \end{aligned} \tag{7}$$

The network for computing the order function (7) is shown in Fig. 2. This network is denoted as ON_q.

2.2.2 Construction of Selection Network

The function of the selection network is to select from among the elements of input array X the element corresponding to the order fixed by the adjustment input and to transfer it to output. This network is composed of n equality sub-networks, ESN, followed by a linear neuron.

a) *Equality Sub-network*

The equality sub-network ESN determines whether the order of an element is equal or not to a given number, k. The number k ($1 \leq k \leq n$) is fixed via the adjustment input.

The function computed by the equality sub-network ESN is defined as follows:

$$\text{for all } x_i \text{ and } 1 \leq k \leq n \ , eq[ord(x_i, X), k] \stackrel{\text{def}}{=} \begin{cases} 1 & \text{if } ord(x_i, X) = k \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

Proposition 2.3 *For all x_i and $1 \leq k \leq n$,*

$$eq[ord(x_i, X), k] = \Phi_T(-\Phi_T(ord(x_i, X) - k - 1) - \Phi_T(k - ord(x_i, X) - 1))$$

where Φ_T is defined in Eq. (4).

Proof : Three cases can be distinguished:

i) If $ord(x_i, X) = k$, then $\Phi_T(ord(x_i, X) - k - 1) = \Phi_T(k - ord(x_i, X) - 1) = \Phi_T(-1) \stackrel{\text{def}}{=} 0$ and consequently $eq[ord(x_i, X), k] = \Phi_T(0) \stackrel{\text{def}}{=} 1$.

ii) If $ord(x_i, X) < k$, then $ord(x_i, X) - k \leq -1$ and $k - ord(x_i, X) \geq 1 \Longrightarrow$ $\Phi_T(ord(x_i, X) - k - 1) = 0$, $\Phi_T(k - ord(x_i, X) - 1) = 1$ and $-\Phi_T(ord(x_i, X) - k - 1) - \Phi_T(k - ord(x_i, X) - 1) = -1$ and consequently $eq[ord(x_i, X), k] = \Phi_T(-1) \stackrel{\text{def}}{=} 0$.

iii) If $ord(x_i, X) > k$, then $ord(x_i, X) - k \geq 1$ and $k - ord(x_i, X) \leq -1 \Longrightarrow$ $\Phi_T(ord(x_i, X) - k - 1) = 1$, $\Phi_T(k - ord(x_i, X) - 1) = 0$ and $-\Phi_T(ord(x_i, X) -$

$k-1) - \Phi_T(k - ord(x_i, X) - 1) = -1$ and consequently $eq[ord(x_i, X), k] = \Phi_T(-1) \stackrel{\text{def}}{=} 0$.
To summarize:
If $ord(x_i, X) = k$, then $eq[ord(x_i, X), k] = 1$, otherwise $eq[ord(x_i, X), k] = 0$.

The equality network EN consists of three threshold-logic neurons as shown by the diagram in Fig.3.

b) *Selection Network*

The selection network is shown in Fig.4 and is denoted as SN it detects and transfers only the appropriate element whose order equals k (i.e., the k^{th} largest element $x_{(k)}$) to output.

Proposition 2.4 *Let $x_{(k)}$ be the k^{th} largest element of the input array X, then:*

$$x_{(k)} = \sum_{i=1}^{n} x_i.eq[ord(x_i, X), k] \tag{9}$$

Proof : Suppose $ord(x_l, X) = k$, then $eq[ord(x_l, X), k] = 1$ and $eq[ord(x_i, X), k] = 0 \ \forall i \in \{1, 2, ..., n\} - \{l\}$. Therefore:

$$\begin{aligned} x_{(k)} &= x_l.eq[ord(x_l, X), k] + \sum_{\substack{i \neq l \\ 1 \leq i \leq n}} x_i.eq[ord(x_i, X), k] \\ &= x_l \times 1 + \sum_{\substack{i \neq l \\ 1 \leq i \leq n}} x_i \times 0 \\ &= x_l \end{aligned}$$

To summarize:
$x_{(k)} = x_l$ if $ord(x_l, X) = k$.

2.2.3 AMAXNET

AMAXNET is shown in Fig.5, where the adjustment input determines which order statistic is to appear at the output. The network illustrated in Fig.5 consists of two kinds of neurons arranged in six layers. The number of neurons in AMAXNET for input size n is $\frac{3}{2}n^2 + \frac{5}{2}n + 1$. There are six layers of neurons in AMAXNET, thus the total processing time is six times the processing time of a single neuron. As the number of elements of the input array increases, only the number of neurons in each layer increases, not the number of layers themselves. Therefore, AMAXNET's total processing time remains constant irrespective of the number of elements in the input array.

2.3 The NNC

In this subsection, our efforts will center on the hardware design of the NNC. First, however, the minimum distance classification (MDC) technique is briefly described.

Consider the classification problem of l classes $C_1, C_2, ..., C_l$ where each pattern class C_j has a reference or templates pattern M^j. An MDC scheme with respect to $M^1, M^2, ..., M^l$ classifies the unknown pattern X to class C_j if $d(X, M^j) = \min_{1 \leq i \leq l}\{d(x, M^i)\}$, where $d(X, M^j)$ is the distance defined between X and M^j. In statistical pattern recognition X is a feature vector; in syntactic pattern recognition X is a structure such as string, tree or graph.

Let $X = (x_1, x_2, ..., x_K)$ and $M^j = (m_1^j, m_2^j, ..., m_K^j)$ be two points in feature space. The distance measure between X and M^j is defined as follows:

$$d(X, M^j) \stackrel{\text{def}}{=} K - \sum_{i=1}^{K} dif[x_i, m_i^j] \tag{10}$$

where

$$dif[x_i, m_i^j] \stackrel{\text{def}}{=} \begin{cases} 1 & \text{if } x_i = m_i^j \\ 0 & \text{otherwise} \end{cases} \tag{11}$$

Proposition 2.5 *Let $X = (x_1, x_2, ..., x_K)$ and $Y = (y_1, y_2, ..., y_K)$ be two elements in K-dimensional space.*

$$\textit{For all } 1 \leq i \leq K, \;\; dif[x_i, y_i] = \Phi_T(-\Phi_Q(x_i - y_i) - \Phi_Q(y_i - x_i)) \tag{12}$$

where Φ_Q and Φ_T are defined respectively in Eq. (3) and Eq. (4).

Proof : Tree cases can be distinguished:

i) If $x_i = y_i$, then $x_i - y_i = y_i - x_i = 0$ and consequently $dif[x_i, y_i] = \Phi_T(0) = 1$ (Cf. Eq. (4)).

ii) If $x_i < y_i$, then $\Phi_Q(x_i - y_i) = 0$ and $\Phi_Q(y_i - x_i) = y_i - x_i$ (Cf. Eq. (3)) and consequently $dif[x_i, y_i] = \Phi_T(0 - (y_i - x_i)) = \Phi_T(x_i - y_i) = 0$ (Cf. Eq. (4)).

ii) If $x_i > y_i$, then $\Phi_Q(x_i - y_i) = x_i - y_i$ and $\Phi_Q(y_i - x_i) = 0$ (Cf. Eq. (3)) and consequently $dif[x_i, y_i] = \Phi_T(-(x_i - y_i) - 0) = \Phi_T(y_i - x_i) = 0$ (Cf. Eq. (4)).

Function (12) is computed by the network shown in Fig.6. This network is denoted as DN (Difference Network).

Function (10) is computed by the distance evaluation network, DEN, illustrated by the diagram in Fig.7.

The classifier NNC is composed of l distance evaluation networks, DENs, already seen above (Cf. Fig.7) and a transference network, which will be

studied below. For the NNC l distances must be calculated between the input feature vector X and l reference vectors $M^1, M^2, ..., M^l$, according to Eq. (10). The minimum distance among the l distances to the input feature vector X, $\min_{1\leq j\leq l}\{d(X, M^j)\}$, must be selected. The index of the reference vector, whose distance to the input feature vector X equals to $\min_{1\leq j\leq l}\{d(X, M^j)\}$, is produced as the class of X:

$$class(X) = q \text{ if } d(X, M^q) = \min_{1\leq j\leq l}\{d(X, M^j)\} \tag{13}$$

where $1 \leq q \leq l$.

Proposition 2.6 *Let $X = (x_1, x_2, ..., x_K)$ and $M^j = (m_1^j, m_2^j, ..., m_K^j)$ be the input feature and the j^{th} reference vectors respectively. The class of X can be evaluated based on:*

$$class(X) = \sum_{p=1}^{l} dif[d(X, M^p), \min_{1\leq j\leq l}\{d(X, M^j)\}].p \tag{14}$$

Proof :
Suppose $d(X, M^q) = \min_{1\leq j\leq l}\{d(X, M^j)\}$, then
$class(X) = q$, $dif[d(X, M^q), \min_{1\leq j\leq l}\{d(X, M^j)\}] = 1$ and
$dif[d(X, M^p), \min_{1\leq j\leq l}\{d(X, M^j)\}] = 0\ \forall p \in \{1, 2, ..., l\} - \{q\}$. Therefore:

$$\begin{aligned}
\sum_{p=1}^{l} dif[d(X, M^p), \min_{1\leq j\leq l}\{d(X, M^j)\}].p &= dif[d(X, M^q), \min_{1\leq j\leq l}\{d(X, M^j)\}].q \\
&\quad + \sum_{\substack{p\neq q \\ 1\leq p\leq l}} dif[d(X, M^p), \min_{1\leq j\leq l}\{d(X, M^j)\}].p \\
&= 1 \times q + \sum_{\substack{p\neq q \\ 1\leq p\leq l}} 0 \times p \\
&= q.
\end{aligned}$$

Proposition 2.7 *Let $X = (x_1, x_2, ..., x_K)$ and $M^j = (m_1^j, m_2^j, ..., m_K^j)$ be the input and the j^{th} reference vectors respectively, where $j \in \{1, 2, ..., l\}$.*

$$d(X, M^q) = \min_{1\leq j\leq l}\{d(X, M^j)\} \Longleftrightarrow$$

$$\forall i \in \{1, 2, ..., K\}, \sum_{p=1}^{l} dif[d(X, M^p), \min_{1\leq j\leq l}\{d(X, M^j)\}].m_i^p = m_i^q \quad where \quad q \in \{1, 2, ..., l\}. \tag{15}$$

Proof :

1. Suppose $d(X,M^q) = \min_{1\leq j\leq l}\{d(X,M^j)\}$, then $dif[d(X,M^q), \min_{1\leq j\leq l}\{d(X,M^j)\}] = 1$ and
$dif[d(X,M^p), \min_{1\leq j\leq l}\{d(X,M^j)\}] = 0 \ \ \forall p \in \{1,2,...,l\} - \{q\}$. Therefore:

$$\begin{aligned}
&\forall i \in \{1,2,...,K\},\\
&\sum_{p=1}^{l} dif[d(X,M^p), \min_{1\leq j\leq l}\{d(X,M^j)\}].m_i^p &= dif[d(X,M^q), \min_{1\leq j\leq l}\{d(X,M^j)\}].m_i^q\\
&&\quad + \sum_{\substack{p\neq q\\ 1\leq p\leq l}} dif[d(X,M^p), \min_{1\leq j\leq l}\{d(X,M^j)\}].m_i^p\\
&&= 1\times m_i^q + \sum_{\substack{p\neq q\\ 1\leq p\leq l}} 0\times m_i^p\\
&&= m_i^q.
\end{aligned}$$

2. Suppose $\forall i \in \{1,2,...,K\}$, $\sum_{p=1}^{l} dif[d(X,M^p), \min_{1\leq j\leq l}\{d(X,M^j)\}].m_i^p = m_i^q$.
Taking into account the fact that $\sum_{p=1}^{l} dif[d(X,M^p), \min_{1\leq j\leq l}\{d(X,M^j)\}] = 1$ and $\forall p \in \{1,2,...,l\}$, $dif[d(X,M^p), \min_{1\leq j\leq l}\{d(X,M^j)\}] \in \{0,1\}$, we have:

$$\begin{aligned}
&\forall i \in \{1,2,...,K\},\\
&\sum_{p=1}^{l} dif[d(X,M^p), \min_{1\leq j\leq l}\{d(X,M^j)\}].m_i^p = \left(\sum_{p=1}^{l} dif[d(X,M^p), \min_{1\leq j\leq l}\{d(X,M^j)\}]\right).m_i^q\\
&\iff \forall i \in \{1,2,...,K\}, \quad \sum_{\substack{p\neq q\\ 1\leq p\leq l}} dif[d(X,M^p), \min_{1\leq j\leq l}\{d(X,M^j)\}].m_i^p\\
&\qquad + \left(dif[d(X,M^q), \min_{1\leq j\leq l}\{d(X,M^j)\}] - \sum_{p=1}^{l} dif[d(X,M^p), \min_{1\leq j\leq l}\{d(X,M^j)\}]\right).m_i^q = 0\\
&\iff \forall i \in \{1,2,...,K\}, \quad \sum_{\substack{p\neq q\\ 1\leq p\leq l}} dif[d(X,M^p), \min_{1\leq j\leq l}\{d(X,M^j)\}].m_i^p\\
&\qquad - \sum_{\substack{p\neq q\\ 1\leq p\leq l}} dif[d(X,M^p), \min_{1\leq j\leq l}\{d(X,M^j)\}].m_i^q = 0\\
&\iff \forall i \in \{1,2,...,K\}, \quad \sum_{\substack{p\neq q\\ 1\leq p\leq l}} dif[d(X,M^p), \min_{1\leq j\leq l}\{d(X,M^j)\}].(m_i^p - m_i^q) = 0\\
&\iff \forall p \in \{1,2,...,l\} - \{q\}, \quad dif[d(X,M^p), \min_{1\leq j\leq l}\{d(X,M^j)\}] = 0.
\end{aligned}$$

As $\sum_{p=1}^{l} dif[d(X,M^p), \min_{1\leq j\leq l}\{d(X,M^j)\}] = 1$, we then have

$$dif[d(X,M^q), \min_{1\leq j\leq l}\{d(X,M^j)\}] = 1,$$

and then:

$$d(X,M^q) = \min_{1\leq j\leq l}\{d(X,M^j)\} \ \text{ (Cf. Eq. (11))}$$

The function of the transference network is to select and transfer to output both the class assigned to the input feature vector X and the reference vector $\widehat{M}$ which satisfies: $d(X, \widehat{M}) = \min_{1 \leq j \leq l}\{d(X, M^j)\}$, where $\widehat{M} \in \{M^1, M^2, ..., M^l\}$.

The transference network, TN, is built by combining an AMAXNET, l difference networks, DNs, and $K+1$ linear neurons as shown in Fig.8. AMAXNET taking as input the l distances to the input feature vector X, $d(X, M^j)$, $j = 1, 2, ..., l$, selects the minimum distance, $\min_{1 \leq j \leq l}\{d(X, M^j)\}$. The minimum distance, $\min_{1 \leq j \leq l}\{d(X, M^j)\}$, thus calculated by AMAXNET is used by the l DNs and $K+1$ linear neurons to identify and transfer to output both the class assigned to the input feature vector X and the reference vector $\widehat{M}$ which satisfies: $d(X, \widehat{M}) = \min_{1 \leq j \leq l}\{d(X, M^j)\}$, where $\widehat{M} \in \{M^1, M^2, ..., M^l\}$.

The network NNC illustrated in Fig.9 is constructed out of three types of neuron the threshold-logic, linear and quasi-linear neurons arranged in 12 layers. Total processing time is constant, irrespective of the number of reference vector, l, and the dimension of the feature vectors, K, and is only 12 times that of a single neuron, in contrast to conventional hardware implementation, where the processing time for classifying an unknown vector is proportional to $(l \times K)$. This not economical when the number of test samples is large, nor acceptable when real-time response is required.

NNC consists of $\frac{3}{2}l^2 + \frac{17}{2}l + K + 3$ neurons. The total number of neurons grows quadratically as the number of reference vectors, l, increases.

3 OCR example

As an example consider the optical character recognition (OCR) problem, with Y a letter in {"a",...,"z"}, and X a feature vector $(x_1, x_2, ..., x_{27})$ whose components are derived from a noisy image of Y. The problem is to recognize a character image as one of the 26 letters, "a",...,"z". Before the features can be calculated, however, some preprocessing of the character images is required.

3.1 Preprocessing

To improve desirable feature vector properties, such as vector similarity within a given class, the images are scaled to remove size variation. The input character image, $f(i, j)$, has 32 rows and 24 columns where i and j are the row and column indexes, respectively. Each pixel is binary-valued, with "0" representing white background and "1" representing part of the printed character. The smallest rectangular subimage is found that contains black pixels. The images are then scaled so that the character fills the 32 by 24 frame, while maintaining a one-to-one height to width perspective.

3.2 Features

Our objective here is to describe the feature set used by the NNC. Our basic approach in constructing a feature set has been to put many types of features into the set. Features 10-27 are developed in William et al. [24].

Consequently, features 1-8, which are vector elements x_1 through x_8, are shown in Table.1 . Feature 9 is the number of components in the character. This helps distinguish "i" and "j" from characters formed by one component. Features 10-13 compute the distances from the four corners of the character image to the first black pixel encountered along the two diagonal lines. This concept is illustrated in Fig.10. Feature 14 is the number of loops in the character. This helps distinguish "a", "b", "d", "e", "o", "p" and "q" from "g" and from nonlooped characters. Features 15 and 16 are the vertical locations of the centers of the loops if they exist. This allows us to distinguish "a" from "e". If there is only one loop, feature 16 is zero. If there are no loops, features 15 and 16 are both zero.

By dividing the character matrix into four disjoint 16 by 12 pixel quadrants, features 17-20 record the maximum horizontal distances from the image sides to a black pixel in the same quadrant. This is indicated in Fig.11.

As shown in Fig.12, feature 21, records how many times column 12 crosses part of the printed character. This helps distinguish "s" and "z" from other characters.

Features 22-24 record the maximum widths of the character in the top five rows, bottom five rows, and middle 10 rows, respectively (Cf. Fig.13). Features 25 and 26 are the height and width of the black portion of the character image.

For rows 20-32, let $r(i)$ denote the column of the rightmost black pixel in row i. Feature 27 equals $max_{20\leq i\leq 32}\{r(i)\} - min_{20\leq i\leq 32}\{r(i)\}$, and is depicted in Fig.14.

The principal objective of this sub-section is to pick a large feature vector X whose elements allow those main local properties to be shown, which differ substantially for different character classes while being similar for different example images of a given class (Cf. Table.2).

3.3 Classifier for OCR example

The neural network classifier to be used for OCR example has 27 inputs, x_1, x_2,..., x_{27}, and one output denoted, O, (Cf. Fig.17).

Let β be a letter in {"a",...,"z"}. The loss function, $l(\beta)$, which measures the loss, or cost, incurred by classifying the image of a character to class β, is defined as follows:

$$l(\beta) = |P(\beta)| - \sum_{p_i \in P(\beta)} p_i \tag{16}$$

where $P(\beta)$ denote the set of main local properties of β (Cf. Table.2), and $p_i \in P(\beta)$ equals "1" when it is true and "0" when it is false. p_i may have three different forms: (i) $p_i = dif[x_i, x_j]$, (ii) $p_i = \Phi_T(x_i - x_j)$ or (iii) $p_i = \Phi_T(-\Phi_T(x_j - x_i))$, where x_i and x_j are two elements of the input feature vector X.

Calculation of the loss function, $l(\beta)$, only necessitates intervention of a limited number of elements of input feature vector X. The network for computing the loss function, $l(\beta)$, is illustrated in Fig.15. This network is denoted as LFEN (Loss Function Evaluation Network).

For example, the network for computing l("b") is constructed out by only using the elements x_7-x_{14}, x_{17}, x_{18}, x_{21}, x_{22} and x_{24}-x_{26}. This network is shown in Fig.16.

Definition 3.1 *Let β be a letter in {"a",...,"z"}. The image of a character is classified to class β if $l(\beta) = \min_{\alpha \in \{\text{"a"},...,\text{"z"}\}} \{l(\alpha)\}$.*

Definition 3.2 *Let β be a letter in {"a",...,"z"}. Let us posit $\alpha_{ord(\beta)} = \beta$, where $ord(\beta)$ denote the order in {"a",...,"z"} of β. The output O is evaluated based on:*

$$O = \sum_{i=1}^{26} i.dif[l(\alpha_i), \min_{1 \le j \le 26} \{l(\alpha_j)\}] \tag{17}$$

where $dif[.,.]$ is the function given by Eqs (11) - (112).

Proposition 3.3 *Let β be a letter in {"a",...,"z"}. The image of a character is classified to class β if $O = ord(\beta)$, where $ord(\beta)$ is the order in {"a",...,"z"} of β.*

Proof :

Suppose $O = ord(\beta)$, then $\exists q \in \{1, 2, ..., 26\}$ such that $q = ord(\beta)$, and $\sum_{i=1}^{26} i.dif[l(\alpha_i), \min_{1 \le j \le 26}\{l(\alpha_j)\}] = q$ (Cf. Eq. (16)).

Taking into account the fact that $\sum_{i=1}^{26} dif[l(\alpha_i), \min_{1 \le j \le 26}\{l(\alpha_j)\}] = 1$ and $\forall i \in \{1, 2, ..., 26\}$,

$dif[l(\alpha_i), \min_{1 \le j \le 26}\{l(\alpha_j)\}] \in \{0, 1\}$, we may write:

$$\sum_{i=1}^{26} i.dif[l(\alpha_i), \min_{1 \le j \le 26} \{l(\alpha_j)\}] = \left(\sum_{i=1}^{26} dif[l(\alpha_i), \min_{1 \le j \le 26} \{l(\alpha_j)\}] \right) .q$$

or, again:

$$\sum_{\substack{i \neq q \\ 1 \le i \le 26}} i.dif[l(\alpha_i), \min_{1 \le j \le 26} \{l(\alpha_j)\}]$$

$$
\begin{aligned}
& +q\left(dif[l(\alpha_q), \min_{1\le j\le 26}\{l(\alpha_j\}] - \sum_{i=1}^{26} dif[l(\alpha_i), \min_{1\le j\le 26}\{l(\alpha_j)\}]\right) = 0 \\
\iff & \sum_{\substack{i\neq q \\ 1\le i\le 26}} i.dif[l(\alpha_i), \min_{1\le j\le 26}\{l(\alpha_j)\}] - q.\sum_{\substack{i\neq q \\ 1\le i\le 26}} dif[l(\alpha_i), \min_{1\le j\le 26}\{l(\alpha_j)\}] = 0 \\
\iff & \sum_{\substack{i\neq q \\ 1\le i\le 26}} (i-q).dif[l(\alpha_i), \min_{1\le j\le 26}\{l(\alpha_j)\}] = 0 \\
\iff & \forall i \in \{1,2,...,26\} - \{q\},\ \ dif[l(\alpha_i), \min_{1\le j\le 26}\{l(\alpha_j)\}] = 0 \\
\iff & dif[l(\alpha_q), \min_{1\le j\le 26}\{l(\alpha_j)\}] = 1 \\
\iff & l(\alpha_q) = \min_{1\le j\le 26}\{l(\alpha_j)\} \\
\iff & l(\alpha_{ord(\beta)}) = \min_{1\le j\le 26}\{l(\alpha_j)\} \\
\iff & l(\beta) = \min_{1\le j\le 26}\{l(\alpha_j)\}
\end{aligned}
$$

Therefore, the image of a character is classified to class β based on the fact that: $l(\beta) = \min_{1\le j\le 26}\{l(\alpha_j)\}$ (Cf. Def.3).

For example, the image of a character is classified to class "c" if $O = 3 = ord(\text{"c"})$.

4 Conclusion

We have shown a neural network solution in fixed time for a classification problem. The neural network classifier, NNC, proposed in this paper consists of three types of neurons, the linear, quasi-linear and threshold-logic neurons. Among all the neurons proposed in the literature, they are probably the easiest to implement in hardware. This NNC network has a very simple configuration, which makes it less subject to the problem caused by poor absolute accuracy in setting up the values of the connection weights. Furthermore, NNC architecture is regular and simple: the connection strengths between the neurons are all fixed, and most of them are just +1 or -1. Therefore, this will greatly facilitate actual hardware implementation of the proposed NNC using currently available VLSI technology.

The OCR example given demonstrates that the approach herein proposed to NNC construction may be profitably applied to solving the classification problem more rapidly and economically than a learning approach.

References

[1] L. R. Bahl, P.F. Brown, P. V. de Souza and R. L. Mercer "A tree-based statistical language model for natural longuage speech recognition", IEEE Trans. Acoust., Speech, Signal Processing, vol. 37, 1001-1008, 1989.

[2] A. Cichoki and R. Unbehauen, "Neural Networks for solving systems of linear equations and related problems", IEEE Trans. Circuits Syst., vol. 39, 124-138, 1992.

[3] L. David and R. W. Andrew, " Optimized feature extraction and the bayes decision in feed-forward classifier networks", IEEE Trans. Pattern Anal. Machine Intell., vol. 13, no. 4, 355-364, 1991.

[4] P. W. Frey and D. J. Slate, " Letter recognition using Holland-style adaptive classifier", Machine Learning, vol. 6, no. 2, 161-182, 1991.

[5] K. Fukushima, ."A neural network for visual pattern recognition", IEEE Computer, pp. 65-75, Mar., 1988.

[6] J. J. Hopfield and D. W. Tank, " Simple 'Neural' optimization networks: An A/D Converter , signal decision circuit, and a linear programming circuit", IEEE Trans. Circuit Syst., vol. CAS-33, 533-541, 1986.

[7] J. J. Hopfield, " Neural networks and physical systems with emergent collective computational ability", Proc. National Academy of Sciences, vol. 79, 2554-2558, 1982.

[8] M. P. Kennedy and L. O. Chua, "Neural networks for nonlinear programming", IEEE Trans. Circuits Syst., vol. 35, 554-562, 1988.

[9] T. Kohonen, "Correlation matrix memories", IEEE Trans. Computers, vol. C-21, 353-359, 1972.

[10] S. Y. Kung and J. S. Taur, "Decision-Based Neural Networks with Signal/Image Classification Applications", IEEE Trans. Neural Networks,

vol. 6, no. 1, 170-181, 1995.

[11] Li Wang and Dong-Chen He, "Texture classification using texture spectrum", Pattern Recognition, vol. 23, no. 8, 905-910, 1990.

[12] R.P. Lippman, "An Introduction to Computing with Neural Nets", IEEE Trans. Accoust., Speech, Signal Processing, vol.35, pp.2-44, 1987.

[13] M. Mestari and A. Namir, "A neural network implementation of L_∞ Metric Partitional Clustering in Fixed Time", SAMS Journal, vol. 41, no. 2, pp. 351-380, 2001.

[14] M. Mestari, A. Namir and J. Abouir, "Switched Capacitor Neural Networks for Optimal Control of Non-linear Dynamic Systems: Design and Stability Analysis", SAMS Journal, vol. 41, no. 3, pp. 559-591, 2001.

[15] M. Mestari and A. Namir, "AOSNET: A neural network implementation of adjustable order statistic filters in fixed time", SAMS Journal, vol. 36, pp. 509-535, 2000.

[16] M. Mestari, "An Analog Neural Network Implementation in Fixed Time of Adjustable Order Statistic Filters and Applications", IEEE Transactions on Neural networks, vol.15, NO. 3, May 2004, pp. 766-785..

[17] M. Mestari and A. Namir, "AMAXNET: A neural network implementation of adjustable MAXNET in fixed time", IFAC-IFIP-IMACS Proc. Internat. Conf. on Control of Industrial Systems, 20-22 May, Belfort, (France), vol. 2, 543-549, 1997.

[18] M. Mestari and A. Namir, "MinMaxNet: A neural network implementation of min/max filters", IFIP Proc. Internat. Conf. on Optimization-based Computer-Aided Modeling and Design, 28-30 May, Noisy-le-grand Paris (France), vol. 1, 26.1-26.4, 1996.

[19] M. Mestari, "$\Theta(1)$ Time Neural Network Minimum Distance Classifier", AMSE International Conference on Modelling and Simulation MS'2000

Las Palmas (Spain).

[20] M. Mestari, "Design of Pattern Cluster Using Neural Networks", AMSE International Conference on Modelling and Simulation MS'2000 Las Palmas (Spain).

[21] A. Chou Philip, "Optimal partitioning for classification and regression trees", IEEE Trans. Pattern Anal. Machine Intell., vol. 13, no. 4, 340-354, 1991.

[22] I. K. Sethi and P. R. Sarvarayudu, "Hierarchical classifier design using mutual information", IEEE Trans. Pattern Anal. Machine Intell., vol. PAMI-4, 441-445, 1982.

[23] J. E. Shore and R. M. Gray, "Minimum cross-entropy pattern classification and cluster analysis", IEEE Trans. Pttern Anal. Machine Intell., vol. PAMI-4, 11-17, 1982.

[24] E. William Weiderman, T. Manry Michael, Hung-chun Yau and Wei Gong, "Comparisons of a neural network and a nearest-neighbor classifier via the numeric handprint recognition problem", IEEE Trans. Neural Networks, vol. 6, no. 6, 1524-1530, 1995.

Received: November 10, 2007

Feature	Possible outcomes	
x_1 (north concavities)	1	noconcavity
	2	shallow concavity
	3	deep concavity
	4	two concavities
	5	three or more concavities
x_2 (south concavities)	1	noconcavity
	2	shallow concavity
	3	deep concavity
	4	two concavities
	5	three or more concavities
x_3 (northwest concavities)	1	noconcavity
	2	northwest concavity
x_4 (northeast concavities)	1	noconcavity
	2	northeast concavity
x_5 (southwest concavities)	1	noconcavity
	2	southwest concavity
x_6 (southeast concavities)	1	noconcavity
	2	southeast concavity
x_7 (vertical bars)	1	novertical bars
	2	one narraow vertical bar
	3	two vertical bars
	4	three or more vertical bars
	5	one wide bar on the right
	6	one wide bar on the left
x_8 (horizontal bars)	1	nohorizontal bars
	2	one horizontal bar
	3	two horizontal bars

TABLE 1 Features 1-8.

Character class	main local properties
"a"	p_1: $x_7 = 1$, p_2: $x_8 = 1$, p_3: $x_9 = 1$, p_4: $x_{14} = 1$, p_5: $x_{15} < \theta_1$, p_6: $x_{21} = 3$, p_7: $x_{25} > x_{26}$, p_8: $x_{27} < x_{26}$
"e"	p_1: $x_7 = 1$, p_2: $x_8 = 1$, p_3: $x_9 = 1$, p_4: $x_{14} = 1$, p_5: $\theta_1 < x_{15}$, p_6: $x_{15} < \theta_2$, p_7: $x_{21} = 3$, p_8: $x_{25} > x_{26}$, p_9: $x_{27} = x_{26}$
"g"	p_1: $x_7 = 1$, p_2: $x_8 = 1$, p_3: $x_9 = 1$, p_4: $x_{14} = 2$, p_5: $x_{15} > \theta_2$, p_6: $x_{16} < \theta_1$, p_7: $x_{21} = 4$, p_8: $x_{25} > x_{26}$
"b"	p_1: $x_7 = 6$, p_2: $x_8 = 1$, p_3: $x_9 = 1$, p_4: $x_{10} < x_{11}$, p_5: $x_{13} = x_{12}$, p_6: $x_{14} = 1$, p_7: $x_{17} < x_{18}$, p_8: $x_{21} = 2$ p_9: $x_{22} < x_{24}$, p_{10}: $x_{25} > x_{26}$
"d"	p_1: $x_7 = 5$, p_2: $x_8 = 1$, p_3: $x_9 = 1$, p_4: $x_{10} > x_{11}$, p_5: $x_{13} = x_{12}$, p_6: $x_{14} = 1$, p_7: $x_{17} > x_{18}$, p_8: $x_{21} = 2$ p_9: $x_{22} < x_{24}$, p_{10}: $x_{25} > x_{26}$
"p"	p_1: $x_7 = 6$, p_2: $x_8 = 1$, p_3: $x_9 = 1$, p_4: $x_{10} = x_{11}$, p_5: $x_{13} < x_{12}$, p_6: $x_{14} = 1$, p_7: $x_{20} < x_{19}$, p_8: $x_{21} = 2$ p_9: $x_{22} > x_{24}$, p_{10}: $x_{25} > x_{26}$
"q"	p_1: $x_7 = 5$, p_2: $x_8 = 1$, p_3: $x_9 = 1$, p_4: $x_{10} = x_{11}$, p_5: $x_{13} > x_{12}$, p_6: $x_{14} = 1$, p_7: $x_{20} > x_{19}$, p_8: $x_{21} = 2$ p_9: $x_{22} > x_{24}$, p_{10}: $x_{25} > x_{26}$
"c"	p_1: $x_7 = 1$, p_2: $x_8 = 1$, p_3: $x_9 = 1$, p_4: $x_{10} = x_{11}$, p_5: $x_{13} = x_{12}$, p_6: $x_{14} = 0$, p_7: $x_{23} < x_{22}$, p_8: $x_{21} = 2$ p_9: $x_{22} \leq x_{24}$, p_{10}: $x_{25} > x_{26}$ p_{11}: $x_6 = 2$ p_{12}: $x_3 = 2$
"r"	p_1: $x_7 = 2$, p_2: x_8 =1 or 2, p_3: $x_9 = 1$, p_4: $x_{10} < x_{11}$, p_5: $x_{12} > x_{13}$, p_6: $x_{14} = 0$, p_7: $x_{21} = 1$ p_8: $x_{22} > x_{24}$, p_9: $x_{25} > x_{26}$, p_{10}: $x_6 = 2$ p_{11}:$x_3 = 1$
"h"	p_1: $x_7 = 6$, p_2: $x_8 = 1$, p_3: $x_9 = 1$, p_4: $x_{10} < x_{11}$, p_5: $x_{17} < x_{18}$, p_6: $x_{14} = 0$, p_7: $x_{21} = 1$ p_8: $x_{22} < x_{24}$, p_9: $x_{25} > x_{26}$, p_{10}: $x_{27} = 0$
"k"	p_1: $x_7 = 6$, p_2: $x_8 = 1$, p_3: $x_9 = 1$, p_4: $x_{10} < x_{11}$, p_5: $x_{17} < x_{18}$, p_6: $x_{14} = 0$, p_7: $x_{21} = 2$ p_8: $x_{22} < x_{24}$, p_9: $x_{25} > x_{26}$, p_{10}: $x_{27} > 0$

Character class	main local properties
"n"	p_1: $x_7 = 3$, p_2: $x_8 = 1$, p_3: $x_9 = 1$, p_4: $x_{14} = 0$, p_5: $x_{21} = 1$, p_6: x_1 =1 or 2, p_7: $x_{25} > x_{26}$
"u"	p_1: $x_7 = 3$, p_2: $x_8 = 1$, p_3: $x_9 = 1$, p_4: $x_{14} = 0$, p_5: $x_{21} = 1$, p_6: $x_1 = 3$, p_7: $x_{25} > x_{26}$
"m"	p_1: $x_7 = 4$, p_2: $x_8 = 1$, p_3: $x_9 = 1$, p_4: $x_{14} = 0$, p_5: $x_{21} = 1$, p_6: x_1 =1 or 2, p_7: $x_{23} \leq x_{22}$, p_8: $x_{23} \leq x_{24}$, p_9: $x_{25} < x_{26}$
"w"	p_1: $x_7 = 1$, p_2: $x_8 = 1$, p_3: $x_9 = 1$, p_4: $x_{14} = 0$, p_5: $x_{21} = 1$, p_6: x_1 =3 or 4 or 5, p_7: $x_2 = 4$, p_8: $x_{23} < x_{22}$, p_9: $x_{23} > x_{24}$, p_{10}: $x_{22} > x_{24}$, p_{11}: $x_{25} < x_{26}$, p_{12}: $x_{17} < x_{20}$
"v"	p_1: $x_7 = 1$, p_2: $x_8 = 1$, p_3: $x_9 = 1$, p_4: $x_{14} = 0$, p_5: $x_{21} = 1$, p_6: x_1 =2 or 3, p_7: $x_{23} < x_{22}$, p_8: $x_{23} > x_{24}$, p_9: $x_{22} > x_{24}$, p_{10}: $x_{25} > x_{26}$, p_{11}: $x_{17} < x_{20}$
"x"	p_1: $x_7 = 1$, p_2: $x_8 = 1$, p_3: $x_9 = 1$, p_4: $x_{14} = 0$, p_5: $x_{21} = 1$, p_6: x_1 =2 or 3, p_7: $x_{23} < x_{22}$, p_8: $x_{23} < x_{24}$, p_9: $x_{22} = x_{24}$, p_{10}: $x_{25} > x_{26}$, p_{11}: $x_{17} = x_{20}$
"y"	p_1: $x_7 = 1$, p_2: $x_8 = 1$, p_3: $x_9 = 1$, p_4: $x_{14} = 0$, p_5: $x_{21} = 1$, p_6: $x_1 = 3$, p_7: $x_{17} \leq x_{20}$, p_8: $x_{22} > x_{24}$, p_9: $x_{25} > x_{26}$
"s"	p_1: $x_7 = 1$, p_2: $x_8 = 1$, p_3: $x_9 = 1$, p_4: $x_{14} = 0$, p_5: $x_{21} = 3$, p_6: $x_{22} = x_{24}$, p_7: $x_{25} > x_{26}$
"z"	p_1: $x_7 = 1$, p_2: $x_8 = 3$, p_3: $x_9 = 1$, p_4: $x_{14} = 0$, p_5: $x_{21} = 3$, p_6: $x_{22} = x_{24}$, p_7: $x_{25} > x_{26}$
"o"	p_1: $x_7 = 1$, p_2: $x_8 = 1$, p_3: $x_9 = 1$, p_4: $x_{14} = 1$, p_5: $x_{21} = 2$, p_6: $x_{23} > x_{22}$, p_7: $x_{23} > x_{24}$, p_8: $x_{25} > x_{26}$, p_9: $x_{15} = \theta_1$
"l"	p_1: $x_7 = 2$, p_2: x_8 =1 or 2, p_3: $x_9 = 1$, p_4: $x_{14} = 0$, p_5: $x_{21} = 1$, p_6: $x_{10} \geq x_{11}$, p_7: $x_1 = 1$, p_8: $x_2 = 1$, p_9: $x_3 = 1$, p_{10}: $x_6 = 1$, p_{11}: $x_{22} \leq x_{24}$, p_{12}: $x_{25} > x_{26}$
"i"	p_1: $x_7 = 2$, p_2: x_8 =1 or 2, p_3: $x_9 = 2$, p_4: $x_{14} = 0$, p_5: $x_{21} = 1$, p_6: $x_{10} = x_{11}$, p_7: $x_1 = 1$, p_8: $x_2 = 1$, p_9: $x_3 = 1$, p_{10}: $x_6 = 1$, p_{11}: $x_{17} \leq x_{18}$, p_{12}: $x_{25} > x_{26}$

Character class	main local properties
"j"	p_1: x_7 =2 or 5, p_2: x_8 =1 or 2, p_3: $x_9 = 2$, p_4: $x_{14} = 0$, p_5: $x_{21} = 1$, p_6: $x_6 = 2$, p_7: $x_{10} > x_{11}$, p_8: $x_{17} > x_{18}$, p_9: $x_{25} > x_{26}$
"t"	p_1: $x_7 = 2$, p_2: $x_8 = 2$, p_3: $x_9 = 1$, p_4: $x_{14} = 0$, p_5: $x_{21} = 2$, p_6: $x_6 = 1$, p_7: $x_{10} < x_{11}$, p_8: $x_{17} < x_{18}$, p_9: $x_{22} < x_{24}$ p_{10}: $x_{25} > x_{26}$
"f"	p_1: $x_7 = 2$, p_2: x_8 =2 or 3, p_3: $x_9 = 1$, p_4: $x_{14} = 0$, p_5: x_{21} =2 or 3, p_6: $x_6 = 2$, p_7: $x_{10} < x_{11}$, p_8: $x_{17} < x_{18}$, p_9: $x_{22} > x_{24}$ p_{10}: $x_{25} > x_{26}$

TABLE.2: The main local porperties of each character class in {"a",...,"z"}, where θ_1 and θ_2 are coefficients chosen by decision-markers (designers), and satisfy: $0 < \theta_1 < \theta_2$. The coefficients θ_1 and θ_2 help to distinguish "e" from "a", and "e" and "a" from "g", respectively.

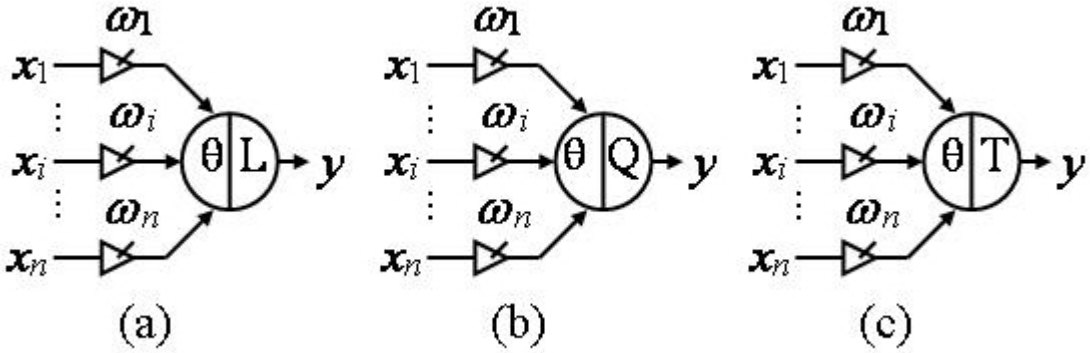

Figure 1: (a) Simplified functional model of an artificial basic neuron cell. (b) Schematic representations of the linear neuron, the quasi-linear neuron, and the threshold-logic neuron respectively. (c) Three possible unipolar transfer characteristics.

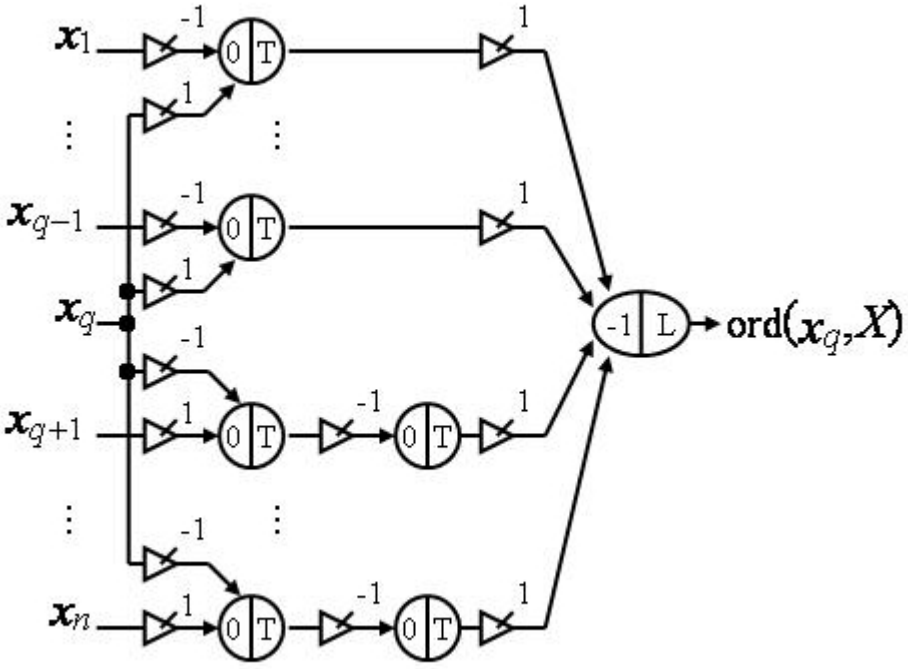

Figure 2: The order network ON_q, $1 \leq q \leq n$.

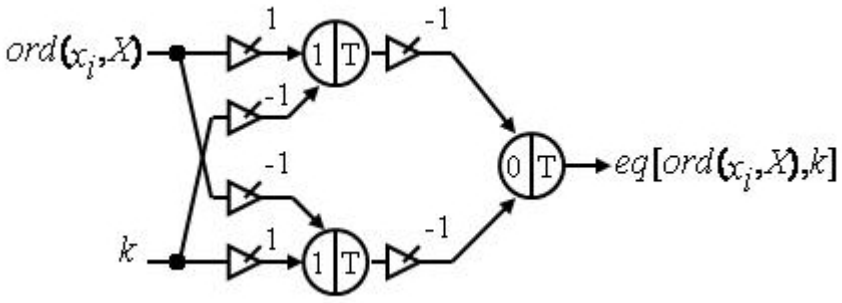

Figure 3: Equality sub-network ESN.

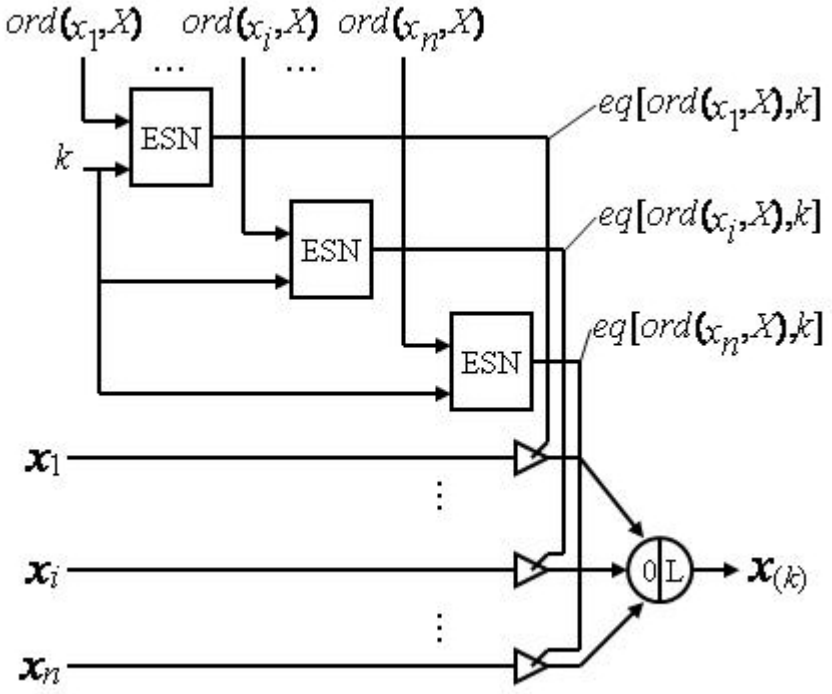

Figure 4: Selection network SN.

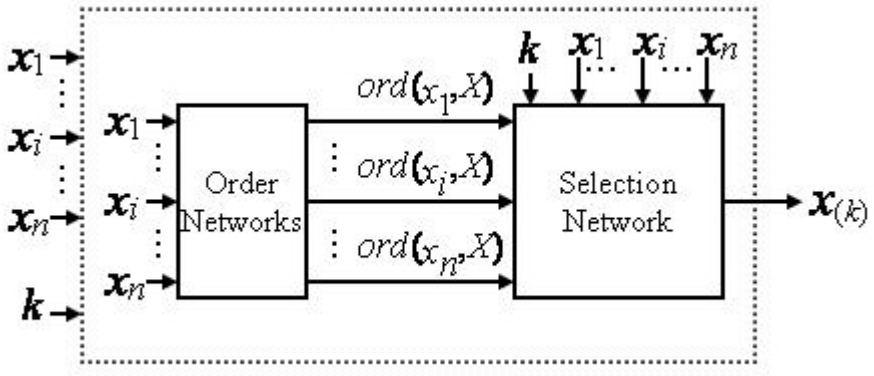

Figure 5: The adjustable MAXNET, AMAXNET.

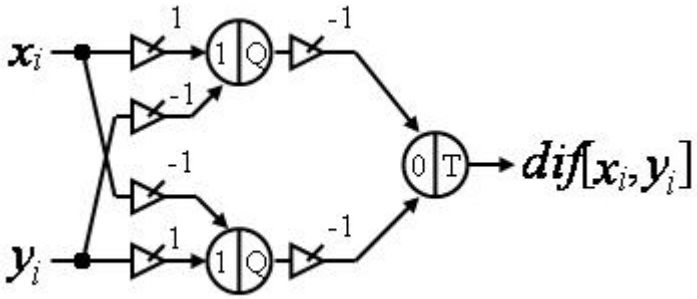

Figure 6: Difference network, DN.

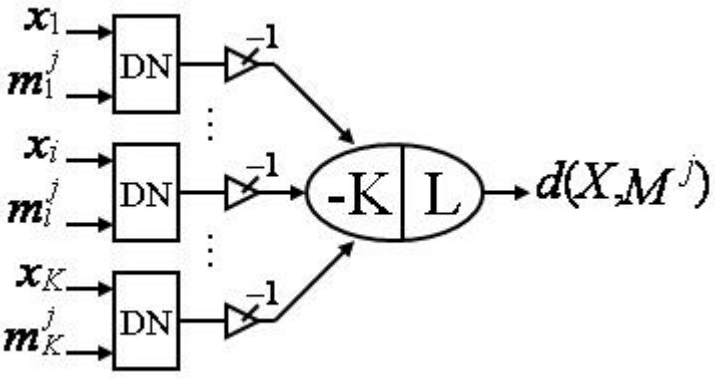

Figure 7: Distance evaluation network, DEN.

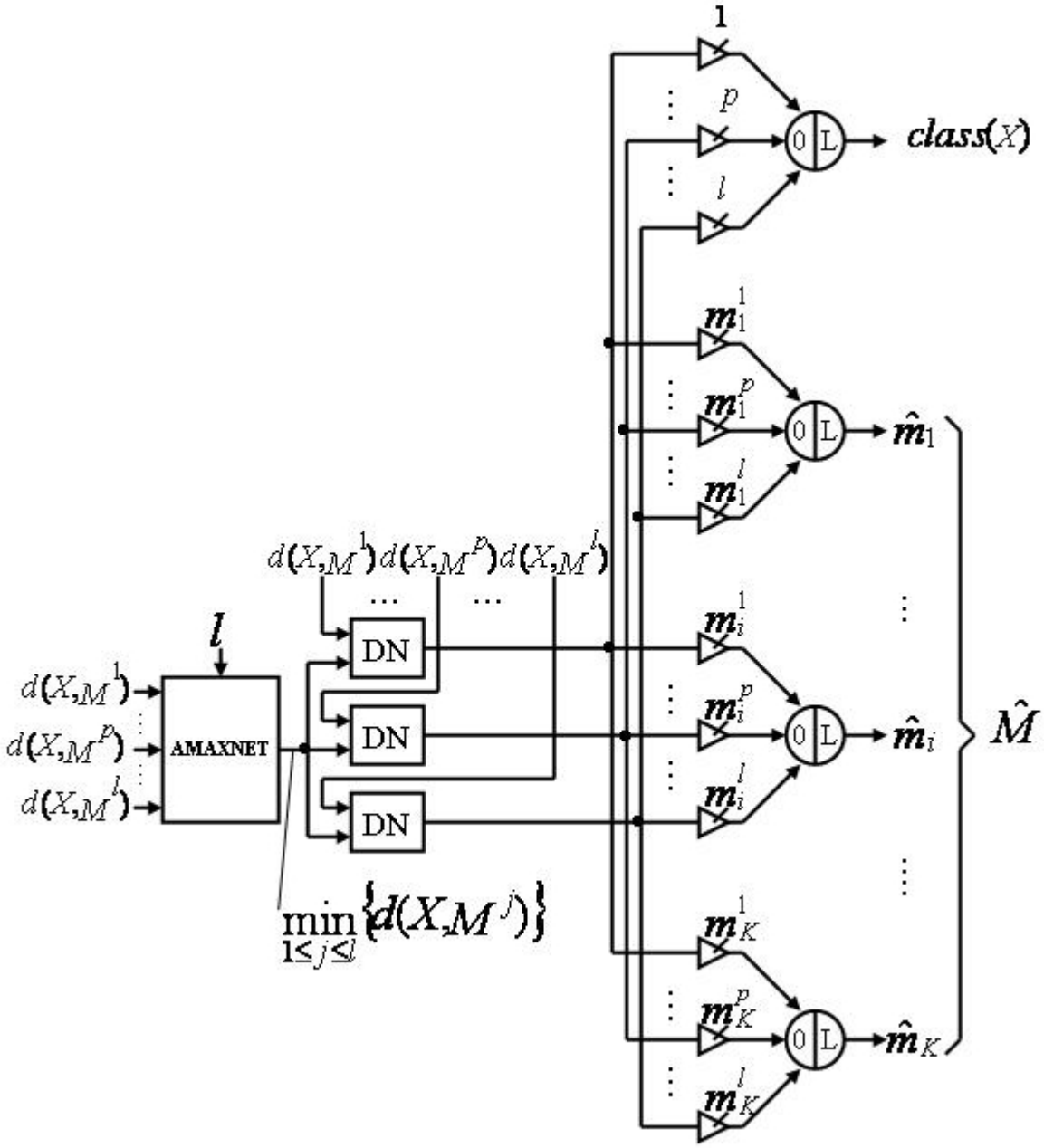

Figure 8: Transference network, TN.

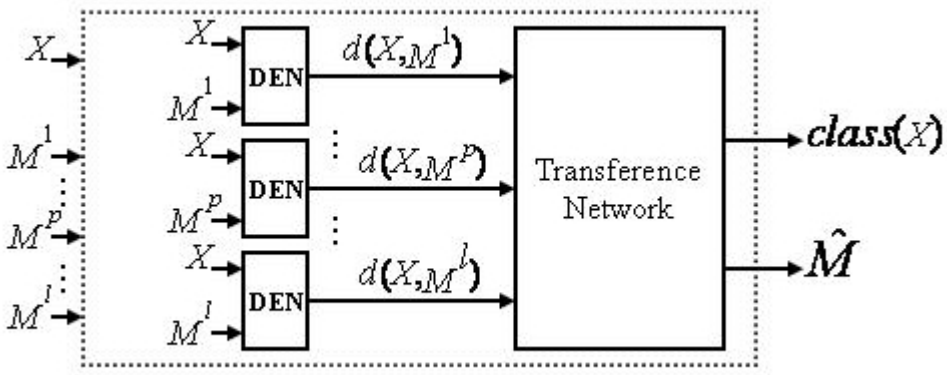

Figure 9: Neural network classifier, NNC.

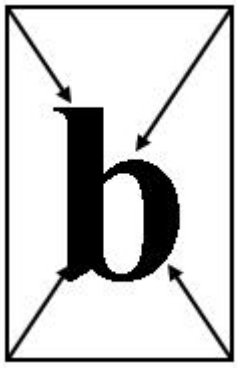

Figure 10: Features 10-13.

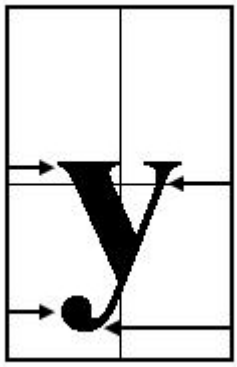

Figure 11: Features 17-20.

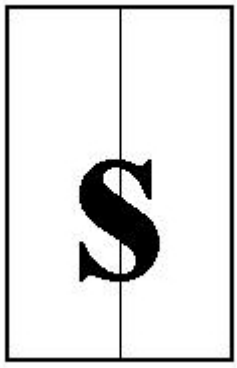

Figure 12: Feature 21.

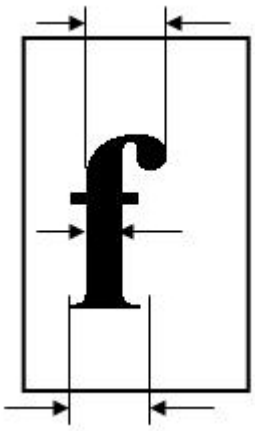

Figure 13: Features 22-24.

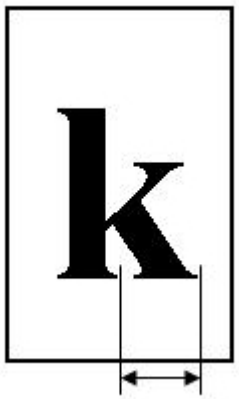

Figure 14: Feature 27

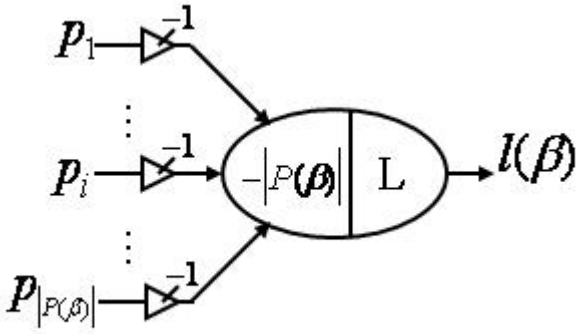

Figure 15: Loss function evaluation network, LFEN.

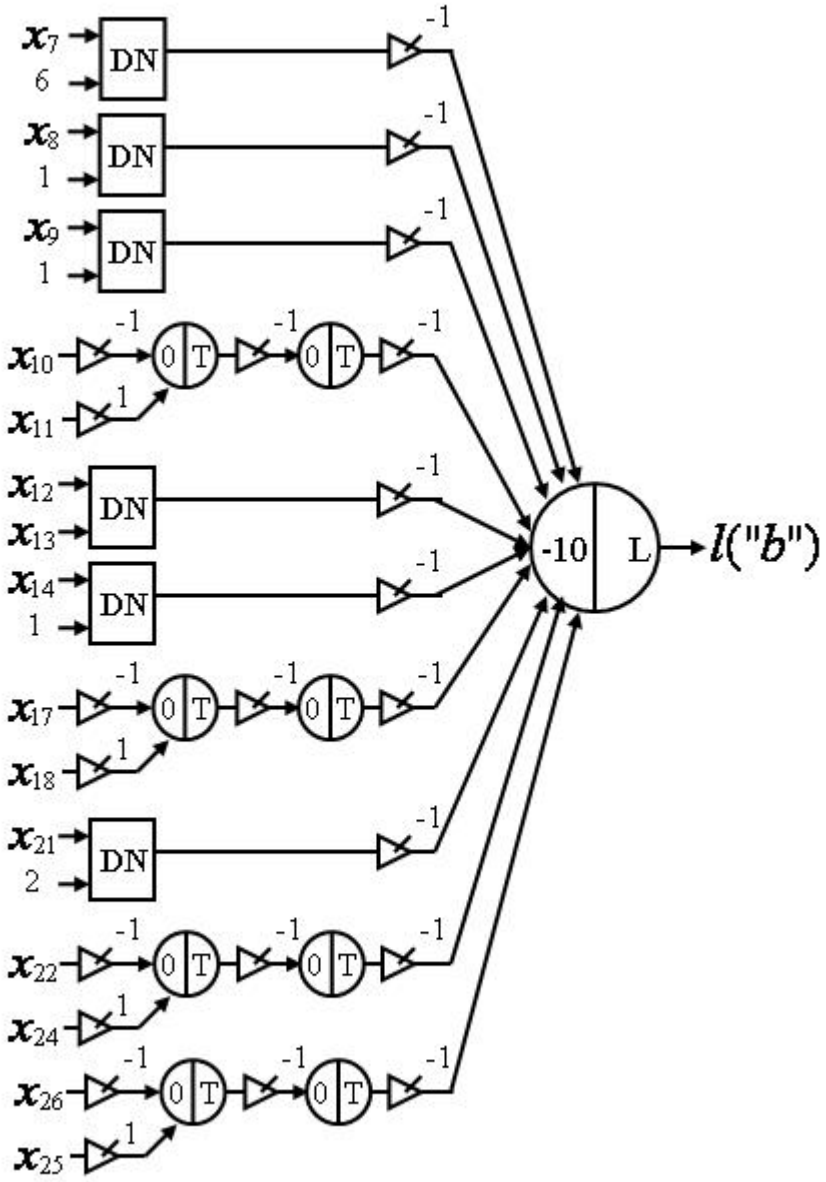

Figure 16: The network for computing the loss function, $l("b")$, where $|P("b")| = 10$ and $P("b") = \{dif[x_7, 6], dif[x_8, 1], dif[x_9, 1], \Phi_T(-\Phi_T(x_{11} - x_{10})), dif[x_{12}, x_{13}], dif[x_{14}, 1], \Phi_T(-\Phi_T(x_{18} - x_{17})), dif[x_{21}, 2], \Phi_T(-\Phi_T(x_{24} - x_{22})), \Phi_T(-\Phi_T(x_{25} - x_{26}))\}$.

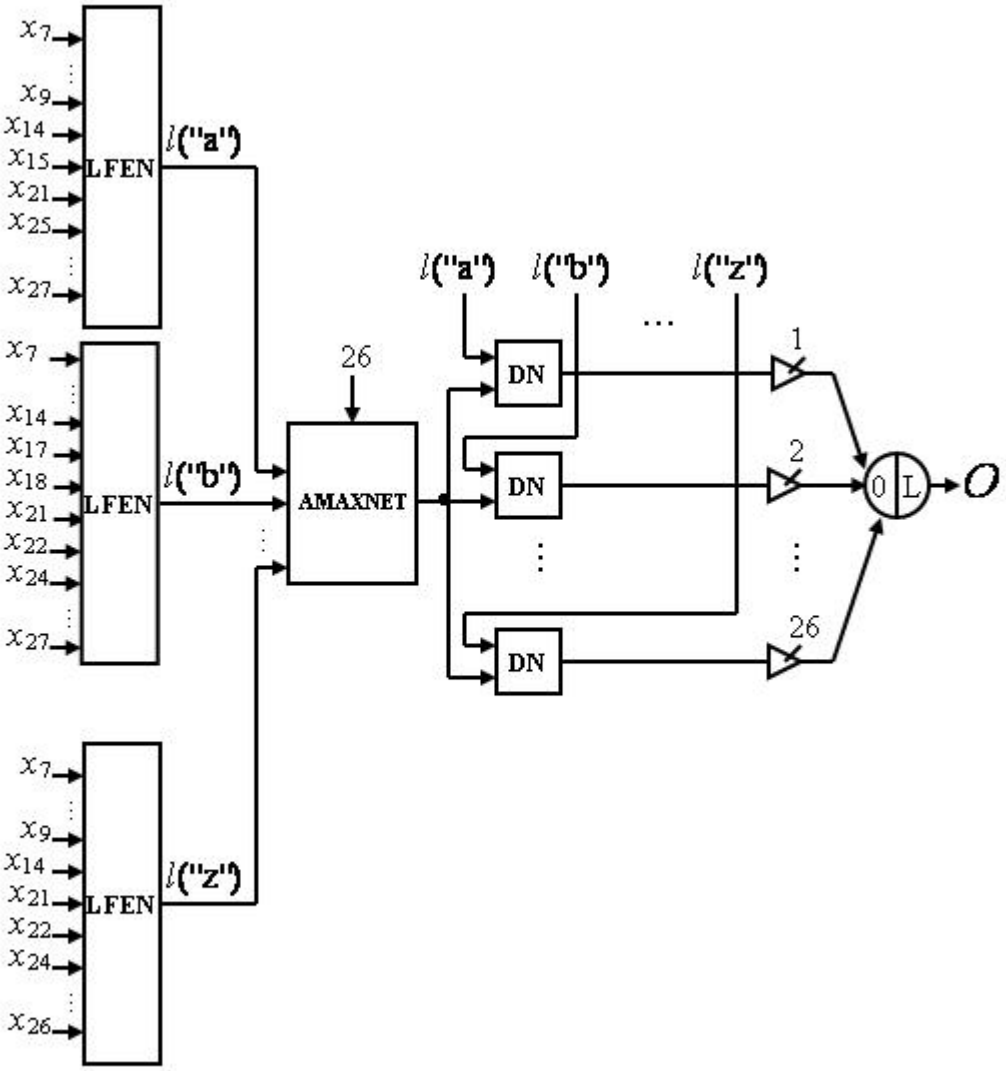

Figure 17: Neural network classifier (NNC) for OCR example.