An Analog Neural Network Implementation in Fixed Time of Adjustable-Order Statistic Filters and Applications

Mohammed Mestari

Abstract—In this paper, we show a neural network implementation in fixed time of adjustable order statistic filters, including sorting, and adaptive-order statistic filters. All these networks accept an array of $N$ numbers $X_i = S_{X_i} M_{X_i} 2^{E_{X_i}}$ as input (where $S_{X_i}$ is the sign of $X_i$, $M_{X_i}$ is the mantissa normalized to $m$ digits, and $E_{X_i}$ is the exponent) and employ two kinds of neurons, the linear and the threshold-logic neurons, with only integer weights (most of the weights being just $+1$ or $-1$ and integer threshold. Therefore, this will greatly facilitate the actual hardware implementation of the proposed neural networks using currently available very large scale integration technology. An application of using minimum filter in implementing a special neural network model neural network classifier (NNC) is given. With a classification problem of $I$ classes $C_1, C_2, \ldots, C_I$, NNC classifies in fixed time an unknown vector to one class using a minimum-distance classification technique.

Index Terms—Adjustable-order statistic filters (AOSFs), minimum-distance classification (MDC) technique, neural networks, neural network classifier (NNC), sorting filter.

I. INTRODUCTION

ORDER statistic filtering is a technique extensively used in pattern recognition and image processing (PRIP) applications [4], [9], [11], [12], [38], [49]–[52], [55], [58], [65], [68], [71], [79]. During the past decades, considerable efforts have been devoted to developing special computer architectures for PRIP applications [7], [41], [42], [54], [63], [66], [69]. Recent advances in very large scale integration (VLSI) microelectronic technology have triggered the idea of implementing PRIP algorithms directly in specialized hardware chips. Many attempts have been made to develop special VLSI devices for such purposes. It is of certain importance and interest to develop a hardware model of high processing speed that can be used as a building block for implementing any order statistic filter (OSF), including sorting and adaptive OSFs (called comparison and selection filters [38]). The main task of the order statistic filtering is to find the $k$th-order statistic of an input array, defined as being the $k$th largest element in the array. This technique finds application in telecommunications, particularly for controlling data packet switches [8], [10]–[13]. In [34], a member of the OSF family shows applications in VLSI auditory and visual systems, while in [20], another filter of OSF family as an analog decoder of error-correcting codes is proposed. OSFs are mostly implemented in software [4], [6], [19], [24], [28], [30], [43], [56], [59], [63], [71], [76], [79], whereas hardware implementations are designed only for specific members, particularly the median and maximum filters, of the OSF family [18], [29], [49], [50], [52], [62], [67], [86].

In this paper, we propose a neural network model, the adjustable order statistic filter (AOSF), which is to be used as a building block for implementation of any member of OSF family. The function of the AOSF is to find in fixed time the $k$th largest element of an array of $N$ numbers $X_i = S_{X_i} M_{X_i} 2^{E_{X_i}}$, where $S_{X_i}$ is the sign of $X_i$ and is coded on 1 bit ($S_{X_i} = 0$ if $X_i$ is positive or zero, and $S_{X_i} = 1$ otherwise); $M_{X_i}$ is the mantissa normalized to $m$ digits and is coded on $m$ bits; and $E_{X_i}$ is the exponent and is coded on $p$ bits; and $i$, $k$ are integers, so that $1 \leq i \leq N$ and $1 \leq k \leq N$.

The approach behind constructing AOSF differs from the conventional approach used in the field of neural networks [33], [81]–[85]. To solve a specific problem, instead of taking a general-purpose network and applying it by learning, we tailor-make a dedicated network. Our primary concern is how to organize neurons into a network so that it can solve a specific problem, with an emphasis on fully utilizing the massive parallelism property offered by neural networks. This approach is useful for solving problems where exact analytic solutions are known or derivable. Two types of neurons are employed in AOSF (linear and threshold-logic neurons). Both types have already been implemented in the past using analogue electronic circuits. They both assume the well-known linear sum neuron model and differ only in their activation functions. A typical implementation of the linear neuron uses a linear operational amplifier [25]. The linear neuron may also be implemented using summing amplifiers (adders) [17] by restricting the value range of the inputs to be within the linear range of the amplifier.

As for the threshold-logic neuron, many different schemes of implementation have been reported. All such neurons have been used in constructing various kinds of neural networks [3], [17], [21], [26], [32], [44], [49]–[53].

An important application of using AOSF ($k = N$) for implementing a special neural network model neural network classifier (NNC) is described. The NNC performs classification using minimum-distance technique. Minimum-distance classification (MDC) is a simple yet powerful technique widely used in statistical recognition, clustering, and other applications.

All neural networks considered in this paper have a feed-forward structure with two kinds of neurons, linear and threshold-
logic neurons. These networks have a very simple configuration, the connection strengths between the neurons are all fixed, most of them being just +1 or −1, which makes hardware implementation easy and straightforward. The modularity and the regularity of the networks’ architecture make them suitable for VLSI implementation.

The processing time of each network herein proposed is constant. As the size of the input array increases, only the number of neurons in each layer increases, not the number of layers themselves. Therefore, the network’s total processing time remains constant, irrespective of the size of the input array. This is in contrast with conventional digital hardware implementation, where the processing time increases along with the input size. Although constant processing time is achievable using unlimited fan-in logic gates, the circuit size grows exponentially as the size of the input increases [15], [23]. The circuit size of AOSF, however, only grows quadratically.

The rest of the paper is organized as follows. In Section II, we describe the construction of the basic neural network AOSF. Section III is devoted to the application of the AOSF in implementation of various order statistic filters, including sorting and adaptive order statistic filters. An important example of application of AOSF \( (k = N) \) to implementing an NNC is given at the end of this section. Section IV contains the conclusion.

II. CONSTRUCTION OF THE AOSF

In this section, we develop the basic neural network, AOSF, which is essential for construction of all neural networks herein proposed, but first, we describe the neurons employed here.

A. Neurons Used

AOSF employs two kinds of neurons, both of which are commonly used in neural network applications [21], [32], [44], [49]–[53]. The only difference between the two is in their activation function: one employs the linear activation function and the other the threshold-logic activation function. Their schematic representations are shown in Fig. 1(b), where \( y \) is the output.
These two kinds of neurons sum the $n$ weighted inputs and pass the result through a nonlinearity according to

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

where $\Phi$ is a limiting or nonlinear transfer characteristic, called an activation function; $\theta (\theta \in \mathbb{R})$ is the external threshold, also called an offset or bias; $\omega_i$ are the synaptic weights or strengths; $x_i$ are the inputs ($i = 1, 2, \ldots, n$), $n$ is the number of inputs, and $y$ represents the output [cf. Fig. 1(a)].

The threshold-logic neuron model [see Fig. 1(b)] uses only the binary (hardlimiting) function [see Fig. 1(c)]. In this model, a weighted sum of all inputs is compared with a threshold $\theta$. If this sum exceeds the threshold, the neuron output is set to “high value” or to “low value” according to

$$\Phi_T(x) \triangleq \begin{cases} 1, & \text{if } x \geq 0 \\ 0, & \text{otherwise} \end{cases}$$  \hfill (2)

where $x = \sum_{i=1}^{n} \omega_i x_i - \theta$, and $\Phi_T$ is the threshold-logic activation function or binary activation function [cf. Fig. 1(c)].
In the case of the linear activation function [see Fig. 1(b)], the output $y$ is given by

$$\Phi_L(x) \overset{\text{def}}{=} x$$

(3)

where $\Phi_L$ is the linear activation function [see Fig. 1(c)] defined by $\Phi_L(x) = x$ with $x = \sum_{i=1}^{n} \omega_i x_i - \theta$.

Both kinds of neurons, threshold and linear, have already been implemented in the past using analog electronic [17], [25].

Fig. 2(a) shows an example of electronic implementation of a linear neuron by employing resistors and operational amplifiers. The output voltage $y_L$ can vary from $-V_{cc}$ to $+V_{cc}$. Applying Kirchhoff’s current law (KCL), we obtain

$$y_L = \sum_{i=1}^{n} \left( \pm \frac{R}{R_i} \right) x_i - \frac{R}{R_{\text{ref}}} V_{\text{ref}}$$

(4)

with the voltage constraints

$$|y_L| \leq +V_{cc}.$$  

(5)

Equation (4) can be written in the compact form

$$y_L = \sum_{i=1}^{n} \omega_i x_i - \theta$$

(6)

where $\omega_i = (\pm R/R_i)$ and $\theta = (R/R_{\text{ref}}) V_{\text{ref}}$.

A model of the threshold-logic neuron can be built using traditional electronic circuit components as shown in Fig. 2(b). The comparator output voltage $y_T$ replaces the output signal of real neuron. The threshold-logic activation function is naturally provided by the saturating characteristic of the amplifier used as comparator. By applying KCL, we obtain the expression

$$y_T = \begin{cases} +V_{cc}, & \text{if } \sum_{i=1}^{n} \left( \pm \frac{R}{R_i} \right) x_i - \left( \frac{R}{R_{\text{ref}}} \right) V_{\text{ref}} \geq 0 \\ 0, & \text{otherwise} \end{cases}$$

(7)

This can be written in the compact form

$$y_T = \begin{cases} +V_{cc}, & \text{if } x \geq 0 \\ 0, & \text{otherwise} \end{cases}$$

(8)

where $x = \sum_{i=1}^{n} \omega_i x_i - \theta$ with $\omega_i = (\pm R/R_i)$ and $\theta = (R/R_{\text{ref}}) V_{\text{ref}}$.

A series of tests carried out on the electronic circuits in Fig. 2(a) and (b) (see Table I) has enabled tracing of linear and threshold-logic transfer characteristics [see Fig. 3(a) and (b)]. Passage from 0 to $+V_{cc}$ of the curve representing the output comparator voltage $y_T$ according to the weighted sum of input signals $x_i (i = 1, 2, \ldots, n), x = \sum_{i=1}^{n} \omega_i x_i - \theta$ is not instantaneous, as foreseen by (8) and the definition of a threshold-logic neuron; this is essentially due to the hysteresis (positive feedback) and the slew rate (13 V/μs) of the op-amplifier (μA741) used as comparator, and also to the gain-bandwidth of op-amplifiers placed before the comparator. We can remedy this problem by using a current feedback op-amplifier whose slew rate reaches (1000 V/μs), such as AD844 [73].

Current-mode signal processing offers several advantages when used in neural circuits. One of the most obvious advantages is that the summing of many signals is most readily accomplished when those signals are currents. Other advantages are increased dynamic range in future VLSI technologies, which are expected to see power supply reductions, high-speed signaling at low impedance nodes due to minimal capacitive

<table>
<thead>
<tr>
<th>Weighted sum of input signals, $x$</th>
<th>Linear neuron output voltage $y_L$</th>
<th>Threshold-logic neuron output voltage $y_T$</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.5468</td>
<td>4.54</td>
<td>4.61351</td>
</tr>
<tr>
<td>4.5968</td>
<td>4.60</td>
<td>4.61352</td>
</tr>
<tr>
<td>4.64548</td>
<td>4.64</td>
<td>4.61353</td>
</tr>
<tr>
<td>4.51857</td>
<td>4.52</td>
<td>4.61351</td>
</tr>
<tr>
<td>3.92399</td>
<td>3.93</td>
<td>4.61345</td>
</tr>
<tr>
<td>2.44654</td>
<td>2.46</td>
<td>4.61327</td>
</tr>
<tr>
<td>0.7389</td>
<td>0.752</td>
<td>4.61307</td>
</tr>
<tr>
<td>0.53596</td>
<td>0.547</td>
<td>4.61304</td>
</tr>
<tr>
<td>0.14688</td>
<td>0.157</td>
<td>4.61274</td>
</tr>
<tr>
<td>0.10087</td>
<td>0.111</td>
<td>4.61207</td>
</tr>
<tr>
<td>0.01013</td>
<td>0.0203</td>
<td>4.60012</td>
</tr>
<tr>
<td>-0.13846</td>
<td>-0.129</td>
<td>2.77385</td>
</tr>
<tr>
<td>-0.21054</td>
<td>-0.201</td>
<td>0.79898</td>
</tr>
<tr>
<td>-0.27743</td>
<td>-0.268</td>
<td>0.39852</td>
</tr>
<tr>
<td>-0.34042</td>
<td>-0.331</td>
<td>0.38817</td>
</tr>
<tr>
<td>-0.40973</td>
<td>-0.401</td>
<td>0.38727</td>
</tr>
<tr>
<td>-0.48818</td>
<td>-0.479</td>
<td>0.38701</td>
</tr>
<tr>
<td>-0.56451</td>
<td>-0.556</td>
<td>0.38709</td>
</tr>
<tr>
<td>-0.71071</td>
<td>-0.703</td>
<td>0.38711</td>
</tr>
<tr>
<td>-0.97613</td>
<td>-0.969</td>
<td>0.3871</td>
</tr>
<tr>
<td>-1.39248</td>
<td>-1.39</td>
<td>0.38709</td>
</tr>
<tr>
<td>-1.75729</td>
<td>-1.76</td>
<td>0.38709</td>
</tr>
</tbody>
</table>
charging/discharging, and extension of linear ranges in transistor circuits. The amount of linearity can be increased by representing signals as current differences in transistors and canceling common-mode nonlinear terms at virtual short inputs of operational transconductance amplifiers. Increased linearity is achieved using more complex cells.

In what follows, our primary concern is how to organize the linear and threshold-logic neurons into a network so that it can solve a specific problem, with an emphasis on full utilization of the massive parallelism property offered by neural networks.

B. Some Basic Functions

In this subsection, we introduce special functions that are essential for construction. First, however, we give the representation employed to represent the elements of input array \( X \).

With a view to making the neural models proposed in this paper adaptable and to then facilitate their incorporation into digital calculators, we will employ the coding used in the majority of present-day computers to represent the elements of input array \( X \).

Let \( X_i \) be an element of the input array \( X(i = 1, 2, \ldots, N) \).

Each element \( X_i \) of \( X \) is represented in a single way by the triplet \((S_{X_i}, M_{X_i}, E_{X_i})\), as follows:

\[
X_i = S_{X_i} M_{X_i} 2^{E_{X_i}} \tag{9}
\]

where

\- \( S_{X_i} \) designating the sign of \( X_i \) is coded on 1 bit \((S_{X_i} = 0 \text{ if } X_i \text{ is positive or zero, and } S_{X_i} = 1 \text{ otherwise})\);

\- \( M_{X_i} \) designating the mantissa normalized to \( m \) digits is coded on \( m \) bits \((M_{X_i} \) is a real number: \((1/2) \leq M_{X_i} < 1\));

\- \( E_{X_i} \) designating the exponent is coded on \( p \) bits \((E_{X_i} \) is a positive, negative, or zero integer).

The mantissa normalized to \( m \) digits \( M_{X_i} \) is represented in (binary) base 2 by

\[
M_{X_i} = \sum_{j=1}^{m} M^j_{X_i} 2^{-j} \tag{10}
\]

where \( M^j_{X_i} (j = 1, 2, \ldots, m) \) are the digits of mantissa \( M_{X_i} \) in base 2. \( M^j_{X_i} \in \{0, 1\} \) for \( 1 \leq j \leq m \) and \( 1 \leq i \leq N \).

Agreeing that \((1/2) \leq M^1_{X_i} \) implies \( M^1_{X_i} \neq 0 \), i.e., \( M^1_{X_i} = 1 \).

The exponent \( E_{X_i} \) is coded in the form “arithmetic complemented to \( 2^n \)” (necessary to encode the negative exponent)

\[
E_{X_i} = \sum_{j=0}^{p-2} E^j_{X_i} - 2^{n-1} \tag{11}
\]

where \( S_{E_{X_i}} \) is the sign bit of \( E_{X_i} \) \((S_{E_{X_i}} = 0 \text{ if } E_{X_i} \text{ is positive or zero, and } S_{E_{X_i}} = 1 \text{ otherwise})\), and \( E^j_{X_i} \in \{0, 1\} \) for \( 0 \leq j \leq p - 2 \) and \( 1 \leq i \leq N \). The exponent \( E_{X_i} \), given by (11), may be calculated by a single neuron (cf. Fig. 4).

We can then represent any element \( X_i \) of the input array \( X \) as a \((m + p + 1)\) bit binary number, as follows:

\[
X_i = (S_{X_i}, M^0_{X_i}, M^0_{X_i}, \ldots, M^p_{X_i}, S_{E_{X_i}}, E^0_{X_i}, E^1_{X_i}, \ldots, E^{p-2}_{X_i}) \tag{12}
\]

Definition 1: 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.

If \((S_{X_i} \neq 0 \text{ and } S_{X_q} \neq 0)\) i.e., \( X_q \) and \( X_i \) are not simultaneously positive, then

\[
\text{comp}(X_i, X_q) = \begin{cases} 1, & \text{if } X_q \geq X_i \\ 0, & \text{if } X_q < X_i \end{cases} \tag{13}
\]

and

\[
\text{comp}(X_i, X_q) = \begin{cases} 1, & \text{if } X_q > X_i \\ 0, & \text{if } X_q \leq X_i \end{cases} \tag{14}
\]

If \((S_{X_q} \neq 0 \text{ and } S_{X_i} \neq 0)\) i.e., \( X_q \) and \( X_i \) are not simultaneously positive, then

\[
\text{comp}(X_i, X_q) = \begin{cases} 1, & \text{if } X_q > X_i \\ 0, & \text{if } X_q \leq X_i \end{cases} \tag{15}
\]
and
\[
\text{for } i > q, \quad \text{comp}(X_i, X_q) = \begin{cases} 
1, & \text{if } X_q \geq X_i \\
0, & \text{if } X_q < X_i.
\end{cases}
\] (16)

**Definition 2:** Let \(X_q\) be an element of the input array \(X\). The order in the input array \(X\) of \(X_q\) is defined as
\[
\text{ord}(X_q, X) \overset{\text{def}}{=} \sum_{i \leq q} \text{comp}(X_i, X_q) + \sum_{i > q} \text{comp}(X_i, X_q) + 1.
\] (17)

**Definition 3:** Let \(X_{(k)}\) denote the \(k\)th largest element of the input array \(X\). Let \(X_q\) be an element of the input array \(X\). Then
\[
X_q = X_{(k)} \iff \text{ord}(X_q, X) = k.
\] (18)

The task of finding the \(k\)th largest element of the input array \(X\) can be done in two phases.

1) Compute the order in the input array \(X\) of any element 
\(X_q(\text{for } q = 1, 2, \ldots, N)\).
2) Select and transfer to output the element of the input array \(X\) corresponding to the order \(k\) desired or chosen by decision-makers (designers).

Note that the operation in either of these two phases can be performed in parallel. This is why it is possible to achieve high processing speed by utilizing the massive parallelism of neural networks.

Corresponding to these two phases, the AOSF is composed of \(N\) order networks, a selection network, and an adjustment input, which allows choice of the order \(k\) of the element to be transferred to output.

**Proposition 1:** \(\Phi_T(x - 1) \equiv x\) if \(x \epsilon \{0, 1\}.
\[
\Phi_T(x - 1) = \Phi_T(-1) = 0 \quad \text{[cf. function (2)].}
\]

**Proof:** If \(x = 0\) then \(\Phi_T(x - 1) = \Phi_T(-1) = 0\) [cf. function (2)].

If \(x = 1\), then \(\Phi_T(x - 1) = \Phi_T(0) = 1\) [cf. function (2)].

**Proposition 2:** Let \(\lor\) be an operator of the set \(\mathcal{R} = \{<, >, =, \geq, \leq\}\). Let \(\overline{M}_{X_i}\) and \(\overline{M}_{X_q}\) be two integers associated to \(M_{X_i}\) and \(M_{X_q}\), respectively, according to (19). Then
\[
M_{X_i} \lor M_{X_q} \iff \overline{M}_{X_i} \lor \overline{M}_{X_q}\].
\] (20)

**Proof:** Let \(\lor\) be an operator of \(\mathcal{R}\). If \(M_{X_i} \lor M_{X_q}\), then \(2^{-m}M_{X_i} \lor 2^{-m}M_{X_q} \Rightarrow \overline{M}_{X_i} \lor \overline{M}_{X_q}\) [cf. relationship (19)].

If \(\overline{M}_{X_i} \lor \overline{M}_{X_q}\), then \(2^{-m}\overline{M}_{X_i} \lor 2^{-m}\overline{M}_{X_q} \Rightarrow M_{X_i} \lor M_{X_q}\) [cf. relationship (19)].

The operator does not change when \(M_{X_i}\) and \(M_{X_q}\) are multiplied by the positive number \(2^m\) or when \(\overline{M}_{X_i}\) and \(\overline{M}_{X_q}\) are multiplied by the positive number \(2^{-m}\).

The integer \(\overline{M}_{X_i}\) associated with the mantissa \(M_{X_i}\) according to (19) is calculated by a linear neuron (cf. Fig. 5), all of whose weights are integers, which facilitate its implementation based on VLSI technology.

Let \(|X_i|\) denote the absolute value of \(X_i\). \(|X_i|\) is represented by the given couple \((M_{X_i}, E_{X_i})\) as follows:
\[
|X_i| = M_{X_i}2^{E_{X_i}}.
\] (21)

As in (12), \(|X_i|\) can be represented as an \((m + p)\) bit binary number
\[
|X_i| = \left(M_{X_i}^{0}, M_{X_i}^{1}, \ldots, M_{X_i}^{m}, S_{E_{X_i}}, E_{X_i}^{0}, E_{X_i}^{1}, \ldots, E_{X_i}^{p-2}\right).
\] (22)
Definition 4: Let \( |X_i| \) and \( |X_q| \) be the absolute value of \( X_i \) and \( X_q \), respectively; we say that \( |X_q| > |X_i| \) if and only if

1) \( E_{X_q} > E_{X_i} \) or
2) \( E_{X_q} = E_{X_i} \) and \( M_{X_q} > M_{X_i} \) [or equivalently \( \bar{M}_{X_q} > \bar{M}_{X_i} \) (cf. Proposition 2)].

Proposition 3: Let \( |X_i| \) and \( |X_q| \) be the absolute value of \( X_i \) and \( X_q \), respectively. Then

\[
\text{comp}(|X_i|, |X_q|) = \begin{cases} 
S_1, & \text{if } i < q \\
S_2, & \text{if } i > q
\end{cases}
\]  

(23)

where

\[
S_1 = \Phi_T \left[ \Phi_T \left( E_{X_q} - E_{X_i} - 1 \right) - \Phi_T \left( E_{X_i} - E_{X_q} - 1 \right) 
+ \Phi_T \left( \bar{M}_{X_q} \right) \right] - 1
\]

(24)

\[
S_2 = \Phi_T \left[ \Phi_T \left( E_{X_q} - E_{X_i} - 1 \right) - \Phi_T \left( E_{X_i} - E_{X_q} - 1 \right) 
+ \Phi_T \left( \bar{M}_{X_q} \right) \right] - 1
\]

and \( \Phi_T \) is defined in (2).

Proof: (See Appendix I.)

Fig. 7. Comparison network of \( X_i \) and \( X_q \). 

The network for computing the comparison function of \( |X_i| \) and \( |X_q| \) is illustrated by the diagram depicted in Fig. 6. This network is denoted as the absolute value comparison network (AVCN).

Definition 5: Let \( |X_i| \) and \( |X_q| \) be the absolute value of \( X_i \) and \( X_q \), respectively; we say that \( X_q > X_i \) if and only if

1) \( S_{X_q} = 0 \) and \( S_{X_i} = 1 \) or
2) \( S_{X_q} = 0 \) and \( S_{X_i} = 0 \) and \( |X_q| > |X_i| \) or
3) \( S_{X_q} = 1 \) and \( S_{X_i} = 1 \) and \( |X_q| < |X_i| \).

Proposition 4: Let \( |X_i| \) and \( |X_q| \) be the absolute value of \( X_i \) and \( X_q \), respectively. Then

\[
\text{comp}(X_i, X_q) = \Phi_T \left( -S_{X_i} - S_{X_q} + \text{comp}(|X_i|, |X_q|) - 1 \right)
+ \Phi_T \left( S_{X_q} + S_{X_i} - \text{comp}(|X_i|, |X_q|) - 2 \right)
+ \Phi_T \left( -S_{X_q} + S_{X_i} - 1 \right).
\]

(26)

Proof: (See Appendix II.)

Fig. 6. Neural network, AVCN, for computing the function comparison (17).
and then $\text{CN}(i,q)$, $i \in \{1,2,\ldots,N\} - \{q\}$ is the network depicted in Fig. 7.

$\text{CN}(i,q)k \in \{1,2,\ldots,N\} - \{q\}$, as shown by the diagram in Fig. 8.

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 networks (ENs) and a detection network (DN), which will be studied hereafter.

1) **Equality Network:** The EN 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 $A_k$, according to

$$ k = \sum_{q=0}^{n-1} a^q_{(k)} 2^q \tag{27} $$

where $a^0_{(k)}, a^1_{(k)}, \ldots, a^{n-1}_{(k)}$ is the word of command allowing choice of the order of the element to be sent to output. $a^q_{(k)} \in \{0,1\}$ for $0 \leq q \leq n - 1$ and $1 \leq k \leq N$.

The function computed by the EN is defined as

$$ \text{eq}[\text{ord}(X_i,X),k] \begin{cases} 0, & \text{if } \text{ord}(X_i,X) = k \\ 1, & \text{otherwise.} \end{cases} \tag{28} $$

**Proposition 5:**

$$ \text{eq}[\text{ord}(X_i,X),k] = \Phi_T(\Phi_T(\text{ord}(X_i,X) - k - 1) + \Phi_T(k - \text{ord}(X_i,X) - 1) - 1) $$

for $1 \leq i \leq N$ and $1 \leq k \leq N \tag{29} $$

where $\Phi_T$ is defined in (2).

**Proof:** Three cases can be distinguished.

1) If $\text{ord}(X_i,X) = k$, then $\Phi_T(\text{ord}(X_i,X) - k - 1) = \Phi_T(k - \text{ord}(X_i,X) - 1) = \Phi_T(-1) = 0$ and consequently $\Phi_T(\text{ord}(X_i,X) - k - 1) + \Phi_T(k - \text{ord}(X_i,X) - 1) - 1) = \Phi_T(-1) = 0$.

2) If $\text{ord}(X_i,X) < k$, then $\text{ord}(X_i,X) - k - 1$ and $k - \text{ord}(X_i,X) - 1$ are both less than or equal to $-1$, and

$$ \Phi_T(\text{ord}(X_i,X) - k - 1) = 0 \quad \Phi_T(k - \text{ord}(X_i,X) - 1) = 0 \quad \Phi_T(-1) = 0 \quad \Phi_T(0) = 0. $$

3) If $\text{ord}(X_i,X) > k$, then $\text{ord}(X_i,X) - k - 1$ and $k - \text{ord}(X_i,X) - 1$ are both greater than or equal to $-1$, and

$$ \Phi_T(\text{ord}(X_i,X) - k - 1) = 0 \quad \Phi_T(k - \text{ord}(X_i,X) - 1) = 0 \quad \Phi_T(-1) = 0 \quad \Phi_T(0) = 0. $$

To summarize: $\Phi_T(\Phi_T(\text{ord}(X_i,X) - k - 1) + \Phi_T(k - \text{ord}(X_i,X) - 1) - 1) = 0$ if $\text{ord}(X_i,X) = k$ and $\Phi_T(\Phi_T(\text{ord}(X_i,X) - k - 1) + \Phi_T(k - \text{ord}(X_i,X) - 1) - 1) = 1$, otherwise and consequently $\text{eq}[\text{ord}(X_i,X),k]$ $= \Phi_T(\Phi_T(\text{ord}(X_i,X) - k - 1) + \Phi_T(k - \text{ord}(X_i,X) - 1) - 1) \tag{28}.$

The EN consists of three threshold-logic neurons, as shown by the diagram in Fig. 9.

2) **Detection Network:** The function of the DN is to detect and send to output the $k$th largest element $X_{(k)}$ of input array $X$.

**Proposition 6:** Let the equation at the bottom of the page [cf. relationship (12)] be the $k$th largest element of input array $X$.

Then

$$ S_{X_{(k)}} = f(S_{X_1}, S_{X_2}, \ldots, S_{X_N}) \tag{30} $$

$$ M_{X_{(j)}}^j = f(M_{X_{(j)}}^{j_1}, M_{X_{(j)}}^{j_2}, \ldots, M_{X_{(j)}}^{j_m}), \quad 1 \leq j \leq m \tag{31} $$

$$ S_{E_{X_{(k)}}} = f(S_{E_{X_1}}, S_{E_{X_2}}, \ldots, S_{E_{X_N}}) \tag{32} $$

$$ E_{X_{(k)}}^q = f(E_{X_{(1)}}^q, E_{X_{(2)}}^q, \ldots, E_{X_{(N)}}^q), \quad 0 \leq q \leq p - 2 \tag{33} $$

where

$$ f(b_1, b_2, \ldots, b_N) = \Phi_T(\sum_{i=1}^N \Phi_T(b_i - \text{eq}[\text{ord}(X_i,X),k] - 1) - 1) $$

with $b_i = (S_{X_i} \text{ or } M_{X_{(j)}}^j \text{ or } S_{E_{X_i}} \text{ or } E_{X_{(k)}}^q \text{ or } E_{X_{(j)}}^q \text{ or } E_{X_{(j)}}^q \text{ or } \ldots) \text{ for } i = 1, 2, \ldots, N.$

$$ X_{(k)} \equiv (S_{X_{(k)}}, M_{X_{(k)}}^1, M_{X_{(k)}}^2, \ldots, M_{X_{(k)}}^m, S_{E_{X_{(k)}}}, E_{X_{(k)}}^0, E_{X_{(k)}}^1, \ldots E_{X_{(k)}}^{p-2}) $$
Proof: Suppose that \( \text{ord}(X_i, X) = k \), then \( \text{eq}[\text{ord}(X_i, X), k] = 0 \) and \( \text{eq}[\text{ord}(X_i, X), k] = 1 \forall i \in \{1, 2, \ldots, N\} \). Therefore

\[
f(b_1, b_2, \ldots, b_N) = f_T(\sum_{i=1}^{N} \Phi_T(b_i - \text{eq}[\text{ord}(X_i, X), k] - 1) - 1)
\]

\[
= \Phi_T(\Phi_T(b_i - \text{eq}[\text{ord}(X_i, X), k] - 1) - 1) + \Phi_T(\sum_{i=1}^{N} \Phi_T(b_i - \text{eq}[\text{ord}(X_i, X), k] - 1) - 1)
\]

\[
= \Phi_T(\Phi_T(b_i - 0 - 1) - 1) + \Phi_T(\sum_{i=1}^{N} \Phi_T(b_i - 1 - 1) - 1)
\]

\[
= \Phi_T(b_i - 1) + \Phi_T(0 - 1) \text{ (cf., proposition 1)}
\]

\[
= b_i \text{ [cf., proposition 1 and function (2)]}
\]

and consequently

\[
S_{X_i} = f(S_{X_1}, S_{X_2}, \ldots, S_{X_N}) \\
M_{X_i} = f(M_{X_1}, M_{X_2}, \ldots, M_{X_N}), \text{ for } 1 \leq j \leq m \\
E_{X_i} = f(E_{X_1}, E_{X_2}, \ldots, E_{X_N}), \text{ for } 0 \leq q \leq p - 2.
\]

As \( S_{X_i} = S_{X_{(k)}}, M_{X_i} = M_{X_{(k)}}, \) for \( 1 \leq j \leq m, S_{E_{X_i}} = S_{E_{X_{(k)}}, E_{X_i} = E_{X_{(k)}}}, \) for \( 0 \leq q \leq p - 2 \), we have

\[
S_{X_{(k)}} = f(S_{X_1}, S_{X_2}, \ldots, S_{X_N}) \\
M_{X_{(k)}} = f(M_{X_1}, M_{X_2}, \ldots, M_{X_N}), \text{ for } 1 \leq j \leq m \\
E_{X_{(k)}} = f(E_{X_1}, E_{X_2}, \ldots, E_{X_N}), \text{ for } 0 \leq q \leq p - 2.
\]

Function (30)–(33) are computed by the network shown by the diagram in Fig. 10. The selection network is shown in Fig. 11 and is denoted as \( SN \). It transfers only the appropriate element whose order equals \( k \) to the output.

D. The AOSF

The AOSF is shown in Fig. 12, where the adjustment input \( A_k \) determines which order statistic is to appear at the output. The network illustrated in Fig. 12 consists of two kinds of neurons arranged in 11 layers. The number of neurons in AOSF for input size \( N \) is \( 14N^2 \times ((m + p - 9)N + m + p + 2) \). There are 11 layers of neurons in the AOSF; thus the processing time is 11 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, AOSF’s total processing time remains constant irrespective of the number of element in the input array. This contrasts with conventional hardware implementation of order statistic filters [14], [74], where the processing time increases along with the number of elements.

The claim that the processing speed of AOSF is independent of its input size does not take into account limitations in the hardware implementation. It is based on the assumption that the processing time of a neuron is independent of its input size. This assumption, however, is not true in analog circuits. For instance, as the number of inputs to a neuron increases, the capacitances of the wires that connect these inputs will increase, causing the settling time to the required accuracy to increase. Therefore, the processing speed of the AOSF to some extent depends on the input size. Even with these limitations, however, the processing speed of the AOSF will still be high enough to have the advantage of speed.

Technologies used in AOSF implementation are broadly categorized into silicon [46]–[48], using analog, digital, or
mixed analog/digital integrated circuits, and optical or electro-optical [1], [5]. No matter which medium is used, the performance of the AOSF would inevitably be affected by the current level of the medium’s technology. Here, we address some problems that might be faced when the AOSF is implemented using analog VLSI circuits. Such problems are also common to other neural network models; however, because the AOSF has a simple configuration, its implementation is less affected.

The first problem is that of poor absolute accuracy in setting up the values of the connection weights. This problem does not arise if the AOSF is implemented using monolithic analog VLSI circuits.

Whatever technology is utilized, the AOSF is not affected by this problem, since the AOSF has a very simple configuration, its weights are all fixed, and most of them are just +1 or −1; they can be set simply by connecting the input to the neuron or by inverting the input before connection.

The second problem is due to the saturation characteristics of the amplifiers used in implementing the linear neuron. For some practical applications, this may not be a serious problem. For example, in image processing applications, the input to the AOSF can be easily scaled to fit in the linear range of the amplifiers.

III. APPLICATIONS

This section presents important examples of the extension of the AOSF to sorting and to adaptive order statistic filters. Finally, an important application of minimum filter for implementing a special neural network model, the NNC, is described.

A. Sorting

Sorting has many applications, especially in data analysis and image processing [80]. Sorting an array is equivalent to giving all order statistics of the array and arranging them either in ascending or descending order.

A more efficient implementation of the sorting network is shown in the diagram in Fig. 13(a). This sorting network is equivalent to N AOSFs set up in parallel, whose common module “order networks” has been merged.

Sorting time is fixed and is only 11 times the processing time for a single neuron. Merging the common module “order networks” permits considerable reduction of the size of the sorting network and a gain of approximately N^3 neurons. A detailed account of a similar implementation can be found in [49]–[51].

A second implementation of the sorting network consists of using n separate AOSF networks in parallel, as shown in the diagram in Fig. 13(b). Sorting time is fixed and does not depend on the size of the input; it is the same time taken for processing a single AOSF.

A third implementation is the use of a single AOSF network [cf. Fig. 13(c)]. By changing the value of k from 1 to N, the elements of the sorted array will appear at the output sequentially. The advantage here is that less neurons are needed; the disadvantage is that the sorting time is proportional to the size of the input.

At each clock pulse [cf. Fig. 13(c)], the counter changes state, and k goes from one value to the next. The AOSF finally produces at its output the input array element whose order corresponds to new value of k. The clock frequency must be lower than AOSF processing speed, i.e., lower than (1/(11τ)), where τ is the processing time for a single neuron.

B. Adaptive Order Statistic Filters

This subsection presents an example of implementing in fixed time of a type of adaptive order statistic filter called comparison-and-selection (CS) filter [38]. The output of the CS filter with parameter J at position l for the input \( X_l = (x_{l-s}, \ldots, x_l, \ldots, x_{l+s}) \) is defined as

\[
y_l = \begin{cases} 
    x_l^{(s+1)} & \text{if } u_l \geq x_l^{(s+1)} \\
    x_l^{(s+1)} & \text{otherwise}
\end{cases}
\] (34)

where \( x_l^{(i)} \) is the \( i \)th largest element in the array \( X_l \), \( u_l \) and \( x_l^{(s+1)} \) are the sample mean and median, respectively, and \( J \) is an integer satisfying \( 1 \leq J \leq N \).

Implementation of the CS filter necessitates the calculation of the sample mean \( u_l \) according to

\[
u_l = \frac{1}{2s+1} \sum_{j=-s}^{s} x_j 
\] (35)

where \( x_j = S_{E_{x_j}}M_{E_{x_j}}2^{E_{x_j}} \), as in relationship (9).

Proposition 7: Let \( E_{x_j} \) be the exponent associated with \( x_j(j = l-s, \ldots, l, \ldots, l+s) \) according to (11). The expression \( 2^{E_{x_j}} \) can be evaluated as

\[
2^{E_{x_j}} = (1 - S_{E_{x_j}}) \left( \sum_{i=1}^{2^{r-1}-1} \Phi_T (E_{x_j} - i) 2^i + 1 \right) + S_{E_{x_j}} \left( 1 - \sum_{i=1}^{2^{r-1}} \Phi_T (-E_{x_j} - i) 2^i \right)
\] (36)

where \( \Phi_T \) is defined in (2).

Proof:

1) Suppose \( S_{E_{x_j}} = 0 \) (i.e., \( E_{x_j} \geq 0 \)); then

\[
(1 - S_{E_{x_j}}) \left( \sum_{i=1}^{2^{r-1}-1} \Phi_T (E_{x_j} - i) 2^i + 1 \right) + S_{E_{x_j}} \left( 1 - \sum_{i=1}^{2^{r-1}} \Phi_T (-E_{x_j} - i) 2^i \right)
\]

\[
= \sum_{i=1}^{2^{r-1}-1} \Phi_T (E_{x_j} - i) 2^i + 1.
\]
Fig. 13. (a) Sorting network made up of $N$ AOSF networks in parallel whose common module “order networks” has been merged. (b) Sorting network made up of $N$ separate AOSF networks in parallel. (c) Sequential sorting network made up of a single AOSF network.
Two cases can be distinguished.

(a) \( E_{x_j} = 0 \), then \( \Phi_T(E_{x_j} - i) = 0 \) \( \forall i \in \{1, 2, \ldots, 2^{n-1} - 1\} \), and consequently
\[
\sum_{i=1}^{2^{n-1}-1} \Phi_T(E_{x_j} - i) 2^{i-1} + 1 = 2^{E_{x_j}=0}.
\]

(b) \( E_{x_j} > 0 \), then \( \Phi_T(E_{x_j} - i) = 0 \) \( \forall i > E_{x_j} \), and \( \Phi_T(E_{x_j} - i) = 1 \) otherwise. Therefore
\[
\sum_{i=1}^{2^{n-1}-1} \Phi_T(E_{x_j} - i) 2^{i-1} + 1 = \sum_{i=1}^{E_{x_j}} \Phi_T(E_{x_j} - i) 2^{i-1} + \sum_{i=E_{x_j}+1}^{2^{n-1}-1} \Phi_T(E_{x_j} - i) 2^{i-1} + 1
\]
\[
= \sum_{i=1}^{E_{x_j}} 2^{i-1} + \sum_{i=E_{x_j}+1}^{2^{n-1}-1} 0 \times 2^{i-1} + 1
\]
\[
= \sum_{i=1}^{E_{x_j}} 2^{i-1} + 1
\]
\[
= 2^{E_{x_j}} + 1
\]
\[
= 2^{E_{x_j}}.
\]

2) Suppose \( S_{E_{x_j}} = 1 \) (i.e., \( E_{x_j} < 0 \)); then
\[
(1 - S_{E_{x_j}}) \left( \sum_{i=1}^{2^{n-1}-1} \Phi_T(E_{x_j} - i) 2^{i-1} + 1 \right)
\]
\[
+ S_{E_{x_j}} \left( 1 - \sum_{i=1}^{2^{n-1}-1} \Phi_T(-E_{x_j} - i) 2^{-i} \right)
\]
\[
= 1 - \sum_{i=1}^{2^{n-1}-1} \Phi_T(-E_{x_j} - i) 2^{-i}
\]
\[
= 1 - \sum_{i=1}^{2^{n-1}-1} \Phi_T(-E_{x_j} - i) 2^{-i}
\]
\[
= 1 - \sum_{i=1}^{2^{n-1}-1} \Phi_T(-E_{x_j} - i) 2^{-i}
\]
\[
= 1 - \sum_{i=1}^{2^{n-1}-1} \Phi_T(-E_{x_j} - i) 2^{-i}
\]
\[
= 1 - \sum_{i=1}^{2^{n-1}-1} 2^{-i} - \sum_{i=E_{x_j}+1}^{2^{n-1}-1} 0 \times 2^{-i}
\]
\[
= 1 - \left( 1 - 2^{E_{x_j}} \right)
\]
\[
= 2^{E_{x_j}}.
\]

The expression \( 2^{E_{x_j}} \) can be calculated by the network illustrated in Fig. 14. The difference between the sample mean \( u_l \) and the median \( x_l^{(s+1)} \), \( u_l - x_l^{(s+1)} = (1/(2s+1)) \sum_{q=-s}^{s} x_q - x_l^{(s+1)} \), can be calculated by the network illustrated in Fig. 15.

**Proposition 8:** The function (34) can be evaluated as

\[
y_l = x_l^{(\mu)}
\]

where

\[
\mu = s + 1 + \left( 2 \times \Phi_T(u_l - x_l^{(s+1)}) - 1 \right) \times J
\]
and \( \Phi_T \) is defined in (2).
feature vector; in syntactic pattern recognition, $X$ is a structure such as a string, tree, or graph.

Let $X = (x_1, x_2, \ldots, x_K)$ and $T^j = (t^j_1, t^j_2, \ldots, t^j_K)$ be two points in feature space. The distance measure between $X$ and $T^j$ is defined as

$$d(X, T^j) = K - \sum_{i=1}^{K} d_i(f(x_i, t^j_i))$$  \hspace{1cm} (39)

where

$$d_i(f(x_i, t^j_i)) = \begin{cases} 1, & \text{if } x_i = t^j_i \\ 0, & \text{otherwise} \end{cases}$$ \hspace{1cm} (40)

Proposition 9: Let $\alpha$ and $\beta$ be two integers, as follows:

$$d_i(f(\alpha, \beta)) = \Phi_T(\Phi_T(\alpha - \beta - 1) - \Phi_T(\beta - \alpha - 1))$$ \hspace{1cm} (41)

where $\Phi_T$ is defined in (2).

Proof: Three cases can be distinguished.

1. If $\alpha = \beta$, then $\Phi_T(\alpha - \beta - 1) = \Phi_T(\beta - \alpha - 1) = \Phi_T(0) = 1$.
2. If $\alpha < \beta$, then $\alpha - \beta \leq -1$ and $\beta - \alpha \geq 1$ \implies $\Phi_T(\alpha - \beta - 1) = 0$, $\Phi_T(\beta - \alpha - 1) = 1$, and $\Phi_T(\alpha - \beta - 1) - \Phi_T(\beta - \alpha - 1) = \Phi_T(-1) = 0$.
3. If $\alpha > \beta$, then $\alpha - \beta \geq 1$ and $\beta - \alpha \leq -1$ \implies $\Phi_T(\alpha - \beta - 1) = 1$, $\Phi_T(\beta - \alpha - 1) = 0$ and $\Phi_T(\alpha - \beta - 1) - \Phi_T(\beta - \alpha - 1) = \Phi_T(-1) = 0$.

To summarize: If $\alpha = \beta$, then $\Phi_T(\alpha - \beta - 1) - \Phi_T(\beta - \alpha - 1) = 0$.

Therefore

$$d_i(f(\alpha, \beta)) = \Phi_T(\Phi_T(\alpha - \beta - 1) - \Phi_T(\beta - \alpha - 1)).$$

Functions (41) and (40) may be calculated by the networks illustrated in Figs. 18 and 19, respectively. These networks are denoted as BDN (the basic difference network) and DNN ($x_i, y_i$) (the difference network of $x_i$ and $y_i$), respectively.

Proposition 10: Let $X = (x_1, x_2, \ldots, x_K)$ and $Y = (y_1, y_2, \ldots, y_K)$ be two elements in $K$-dimensional space. Let $(S_{x_i}, M_{x_i}, E_{x_i})$ and $(S_{y_i}, M_{y_i}, E_{y_i})$ be respective representations of $x_i$ and $y_i$ [cf. relationship (9)], as follows:

For all $1 \leq i \leq K$,

$$d_i(f(x_i, y_i)) = \Phi_T(d_i(f(S_{x_i}, S_{y_i}) + d_i(M_{x_i}, M_{y_i}) + d_i(E_{x_i}, E_{y_i}) - 3))$$ \hspace{1cm} (42)

where $\widehat{M}_{x_i}$ and $\widehat{M}_{y_i}$ are the integers associated with $M_{x_i}$ and $M_{y_i}$ according to formula (19), and $\Phi_T$ is defined in (2).

Proof: $\Phi_T(d_i(f(S_{x_i}, S_{y_i}) + d_i(M_{x_i}, M_{y_i}) + d_i(E_{x_i}, E_{y_i}) - 3)) \iff (d_i(S_{x_i}, S_{y_i}) = 1 \text{ and } d_i(\widehat{M}_{x_i}, \widehat{M}_{y_i}) = 1 \text{ and } d_i(E_{x_i}, E_{y_i}) = 1).$ We know the following:

- $d_i(S_{x_i}, S_{y_i}) = 1 \iff S_{x_i} = S_{y_i}$.
- $d_i(\widehat{M}_{x_i}, \widehat{M}_{y_i}) = 1 \iff \widehat{M}_{x_i} = \widehat{M}_{y_i}$, or equivalently $M_{x_i} = M_{y_i}$ (cf. Proposition 2).
- $d_i(E_{x_i}, E_{y_i}) = 1 \iff E_{x_i} = E_{y_i}$.

C. Neural Network Classifier

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

Consider the classification problem of $l$ classes $C_1, C_2, \ldots, C_l$ where each pattern class $C_j$ has a reference or template pattern $T^j$. An MDC scheme with respect to $T^1, T^2, \ldots, T^l$ classifies the unknown pattern $X$ to class $C_j$ if $d(X, T^j) = \min_{1 \leq i \leq l} d(X, T^i)$, where $d(X, T^j)$ is the distance defined between $X$ and $T^j$. In statistical pattern recognition, $X$ is a
Fig. 18. Basic difference network.

Fig. 19. Difference network of \( x_i \) and \( y_i \), \( DN(x_i, y_i) \), enabling calculation of expression (40), where BDN is the network shown in Fig. 18.

To summarize,

\[
\Phi_T \left( \text{dif}(S_{x_i}, S_{y_i}) + \text{dif}(\overline{M}_{x_i}, \overline{M}_{y_i}) + \text{dif}(E_{x_i}, E_{y_i}) - 3 \right) = 1 \ \text{iff} \ (S_{x_i} = S_{y_i} \ \text{and} \ M_{x_i} = M_{y_i} \ \text{and} \ E_{x_i} = E_{y_i})
\]

or again

\[
\Phi_T \left( \text{dif}(S_{x_i}, S_{y_i}) + \text{dif}(\overline{M}_{x_i}, \overline{M}_{y_i}) + \text{dif}(E_{x_i}, E_{y_i}) - 3 \right) = 1 \ \text{iff} \ x_i = y_i
\]

and consequently

\[
\text{dif}(x_i, y_i) = \Phi_T \left( \text{dif}(S_{x_i}, S_{y_i}) + \text{dif}(\overline{M}_{x_i}, \overline{M}_{y_i}) + \text{dif}(E_{x_i}, E_{y_i}) - 3 \right).
\]

Function (39) is computed by the distance evaluation network (DEN) illustrated by the diagram in Fig. 20.

The classifier NNC is composed of \( l \) DENs already seen above (cf. Fig. 20) and a transference network, which will be studied subsequently. For the NNC, \( l \) distances must be calculated between the input feature vector \( X \) and \( l \) reference vectors \( T^1, T^2, \ldots, T^l \), according to (39) and (40). The minimum distance among the \( l \) distances to the input feature vector \( X \), \( \min_{1 \leq i \leq l} \{d(X, T^i)\} \) must be selected. The index of the reference vector, whose distance to the input feature vector \( X \) is equal to \( \min_{1 \leq i \leq l} \{d(X, T^i)\} \), is produced as the class of \( X \), as follows:

\[
\text{class}(X) = q \ \text{if} \ d(X, T^q) = \min_{1 \leq i \leq l} \{d(X, T^i)\}. \quad (43)
\]

Proposition 11: Let \( X = (x_1, x_2, \ldots, x_K) \) and \( T^j = (T^j_1, T^j_2, \ldots, T^j_K) \) be the input feature and the \( j \)th reference vectors, respectively. The class of \( X \) can be evaluated based on

\[
\text{class}(X) = \sum_{p=1}^{l} \text{dif} \left[ d(X, T^p), \min_{1 \leq i \leq l} \{d(X, T^i)\} \right] p. \quad (44)
\]

Proof: Suppose \( d(X, T^q) = \min_{1 \leq i \leq l} \{d(X, T^i)\} \), then
class \((X) = q \) [cf. (43)], \( \text{dif}[d(X, T^q), \min_{1 \leq i \leq l} \{d(X, T^i)\}] = 1 \) and \( \text{dif}[d(X, T^p), \min_{1 \leq i \leq l} \{d(X, T^i)\}] = 0 \ \forall \ p \ \in \{1, 2, \ldots, l\} \setminus \{q\} \). Therefore

\[
\sum_{p=1}^{l} \text{dif} \left[ d(X, T^p), \min_{1 \leq i \leq l} \{d(X, T^i)\} \right] p
\]

\[
= \text{dif} \left[ d(X, T^q), \min_{1 \leq i \leq l} \{d(X, T^i)\} \right] q
\]

\[
+ \sum_{\substack{p\neq q \ 1 \leq p \leq l}} \text{dif} \left[ d(X, T^p), \min_{1 \leq i \leq l} \{d(X, T^i)\} \right] p
\]

\[
= 1 \times q + \sum_{\substack{p\neq q \ 1 \leq p \leq l}} 0 \times p
\]

\[
= q.
\]

Proposition 12: Let \( X = (x_1, x_2, \ldots, x_K) \) and \( T^p = (t^p_1, t^p_2, \ldots, t^p_K) \) be the input and the \( p \)th reference vectors, respectively, where \( p \in \{1, 2, \ldots, l\} \), as follows:

\[
d(X, T^q) = \min_{1 \leq i \leq l} \{d(X, T^i)\} \iff \forall j \in \{1, 2, \ldots, K\},
\]

\[
\sum_{i=1}^{l} \text{dif} \left[ d(X, T^p), \min_{1 \leq i \leq l} \{d(X, T^i)\} \right] t^p_i = t^q_i.
\]

Proof:

1) Suppose \( d(X, T^q) = \min_{1 \leq i \leq l} \{d(X, T^i)\} \), then

\[
\text{dif}[d(X, T^p), \min_{1 \leq i \leq l} \{d(X, T^i)\}] = 1 \text{ and...}
\]
\[
\text{diff}[d(X,T^p), \min_{1 \leq i \leq l} \{d(X,T^i)\}] = 0 \quad \forall p \in \{1, 2, \ldots, l\} - \{q\}. \text{ Therefore}
\]

\[
\forall j \in \{1, 2, \ldots, K\},
\sum_{p=1}^{l} \text{diff} \left[ d(X,T^p), \min_{1 \leq i \leq l} \{d(X,T^i)\} \right] \cdot d_j^p = \text{diff} \left[ d(X,T^q), \min_{1 \leq i \leq l} \{d(X,T^i)\} \right] \cdot d_j^q
\]

\[
+ \sum_{p \neq q} \text{diff} \left[ d(X,T^p), \min_{1 \leq i \leq l} \{d(X,T^i)\} \right] \cdot d_j^p
\]

\[
= 1 \times d_j^q + \sum_{p \neq q} 0 \times d_j^p
\]

\[
= d_j^q.
\]

2) Suppose \( \forall j \in \{1, 2, \ldots, K\}, \sum_{p=1}^{l} \text{diff} \left[ d(X,T^p), \min_{1 \leq i \leq l} \{d(X,T^i)\} \right] \cdot d_j^p = d_j^q \)

Taking into account the fact that

\[
\sum_{p=1}^{l} \text{diff} \left[ d(X,T^p), \min_{1 \leq i \leq l} \{d(X,T^i)\} \right] = 1 \quad \text{and} \quad \forall p \in \{1, 2, \ldots, l\}, \text{diff}[d(X,T^p), \min_{1 \leq i \leq l} \{d(X,T^i)\}] \in \{0, 1\},
\]

we have the equation shown at bottom of page. As \( \sum_{p=1}^{l} \text{diff} \left[ d(X,T^p), \min_{1 \leq i \leq l} \{d(X,T^i)\} \right] = 1 \), we then have

\[
\text{diff} \left[ d(X,T^q), \min_{1 \leq i \leq l} \{d(X,T^i)\} \right] = 1
\]

and then

\[
d(X,T^q) = \min_{1 \leq i \leq l} \{d(X,T^i)\}.
\]

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 \( \hat{T} \in \{T^1, T^2, \ldots, T^l\} \), which satisfies

\[
d(X, \hat{T}) = \min_{1 \leq i \leq l} \{d(X,T^i)\}.
\]

The transference network (TN) is built by combining a minimum filter (i.e., AOSF for \( k = N \), \( l \) DNs, and \( K + 1 \) linear neurons, as shown in Fig. 21. The minimum filter taking as input the \( l \) distances to the input feature vector \( X, d(X,T^i) \), \( i = 1, 2, \ldots, l \), selects the minimum distance \( \min_{1 \leq i \leq l} \{d(X,T^i)\} \). The minimum distance \( \min_{1 \leq i \leq l} \{d(X,T^i)\} \) thus calculated by the minimum filter 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 \( \hat{T} \), which satisfies relationship (45).

The network NNC illustrated in Fig. 22 is constructed out of two types of neurons, threshold-logic and linear neurons, arranged in 19 layers. Total processing time is constant, irrespective of the number of reference vectors \( l \) and the dimension of the feature vectors \( K \) and is only 19 times that of a single neuron, in contrast to conventional hardware implementation, where the processing time for classifying an unknown vector is proportional to \((l \times K)\). Automatic recognition of handwritten numerals and characters has been an active subject of research due to its potential for intelligent man–machine interface, [2], [16], [22], [35]–[37], [39], [40], [45], [57], [60], [64], [70], [72], [75]. Handwritten numeral and character recognition has been computed via either statistical or syntactic approaches. In the statistical approach, a pattern is represented by a set of \( K \)-dimensional feature vectors, and the decision-making
process is determined by a similarity measure such as a distance metric or a discriminant function. NNC can thus be applied very beneficially to the difficult real-world problem of handwritten numeral and character recognition. NNC is computationally attractive when compared with a conventional pattern classifier [78].

IV. CONCLUSION

We have shown a neural network implementation in fixed time of adjustable order statistic filter AOSF. The AOSF is used as a building block for implementing in fixed time all members of the OSF family, including sorting and adaptive order statistic filters. An application of AOSF (for $k = N$) for implementing in fixed time a special neural network model NNC is given.

All neural networks herein proposed have a feed-forward structure and consist of two kinds of neurons—linear and threshold-logic neurons. Among all the neurons proposed in the literature, they are probably the easiest to implement in hardware. These neural networks have a very simple configuration, which makes hardware implementation less subject to any problems caused by poor absolute accuracy in setting up the values of the connection weights. Furthermore, these neural networks’ 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 proposed neural networks using currently available VLSI technology.

APPENDIX I

PROOF OF PROPOSITION 3

1)  
$E_{xi} > E_{xq} \implies E_{xq} - E_{xi} \geq 1$ and $E_{xq} - E_{xi} \geq -1$ (as $E_{xq}$ and $E_{xi}$ are integers) \( \implies \Phi_T(E_{xq} - E_{xi} - 1) = 1 \) and \( \Phi_T(E_{xq} - E_{xi} - 1) = 0 \) [cf. function (2)],

which leads to

$$
\Phi_T(-\Phi_T(E_{xq} - E_{xi} - 1) - \Phi_T(E_{xi} - E_{xq} - 1) + \Phi_T(\hat{M}_{xq} - \hat{M}_{xi} - 1)) \geq 0
$$

and consequently

$s_i = \Phi_T(1) = 1$ [cf. function (2)];

2)  
$E_{xi} < E_{xq} \implies E_{xq} - E_{xi} \geq 1$ and $E_{xq} - E_{xi} \leq -1$ (as $E_{xq}$ and $E_{xi}$ are integers) \( \implies \Phi_T(E_{xq} - E_{xi} - 1) = 0 \) and \( \Phi_T(E_{xq} - E_{xi} - 1) = 1 \) [cf. function (2)],

which leads to

$$
\Phi_T(-\Phi_T(E_{xq} - E_{xi} - 1) - \Phi_T(E_{xi} - E_{xq} - 1) + \Phi_T(\hat{M}_{xq} - \hat{M}_{xi} - 1)) \leq 0
$$

and consequently

$s_i = \Phi_T(-1) = 0$ [cf. function (2)];

3)  
$\forall i < q$

\( E_{xi} > E_{xq} \implies E_{xq} - E_{xi} \geq 1 \) and $E_{xq} - E_{xi} \geq -1$ (as $E_{xq}$ and $E_{xi}$ are integers) \( \implies \Phi_T(E_{xq} - E_{xi} - 1) = 1 \) and \( \Phi_T(E_{xq} - E_{xi} - 1) = 0 \) [cf. function (2)],
which leads to
\[ \Phi_T(-\Phi_T(E_{X_1} - E_{X_2} - 1) - \Phi_T(E_{X_1} - E_{X_2} - 1) + \Phi_T(M_{X_1} - M_{X_2} - 1) - 1) = \Phi_T(\Phi_T(M_{X_1} - M_{X_2} - 1) - 1) \]
(cf. Proposition 1), \( \Phi_T(-\Phi_T(E_{X_1} - E_{X_2} - 1) - \Phi_T(E_{X_1} - E_{X_2} - 1) + \Phi_T(M_{X_1} - M_{X_2} - 1) - 1) = \Phi_T(\Phi_T(M_{X_1} - M_{X_2} - 1) - 1) \) (cf. Proposition 1), and consequently \( \Phi_T(M_{X_1} - M_{X_2} - 1) - 1) = \Phi_T(\Phi_T(M_{X_1} - M_{X_2} - 1) - 1) \) (cf. Proposition 1) and

therefore
\[ S_1 = \Phi_T(M_{X_1} - M_{X_2} - 1) - 1) = \Phi_T(M_{X_1} - M_{X_2} - 1) \]

Three cases may be distinguished as follows:

- If \( M_{X_1} > M_{X_2} \), then \( M_{X_1} > M_{X_2} - 1 \)
- If \( M_{X_1} < M_{X_2} \), then \( M_{X_1} = M_{X_2} - 1 \)
- If \( M_{X_1} = M_{X_2} \), then \( M_{X_1} = M_{X_2} - 1 \)

and consequently
\[ S_1 = \Phi_T(1) = 1 \]

To summarize
\[ S_1 = \begin{cases} 1, & (E_{X_1} > E_{X_2}) \\ 0, & (E_{X_1} = E_{X_2}) \\ 0, & (E_{X_1} = E_{X_2} < M_{X_1}) \end{cases} \]

Taking into account Definition 4 and Proposition 2, we have
\[ S_1 = \begin{cases} 1, & \text{if} \; |X_q| \geq |X_i| \\ 0, & \text{if} \; |X_q| < |X_i| \end{cases} \]

According to Definition 1, we may write
\[ \text{comp}(|X_i|, |X_q|) = S_1 \text{ if } i < q \]

\[ \Phi_T(-\Phi_T(E_{X_1} - E_{X_2} - 1) - \Phi_T(E_{X_1} - E_{X_2} - 1) + \Phi_T(M_{X_2} - M_{X_1} - 1) - 1) = \Phi_T(\Phi_T(M_{X_1} - M_{X_2} - 1) - 1) \]
(cf. Proposition 1), \( \Phi_T(-\Phi_T(E_{X_1} - E_{X_2} - 1) - \Phi_T(E_{X_1} - E_{X_2} - 1) + \Phi_T(M_{X_1} - M_{X_2} - 1) - 1) = \Phi_T(\Phi_T(M_{X_1} - M_{X_2} - 1) - 1) \)
(cf. Proposition 1), and consequently \( \Phi_T(M_{X_1} - M_{X_2} - 1) - 1) = \Phi_T(\Phi_T(M_{X_1} - M_{X_2} - 1) - 1) \)
(cf. Proposition 1) and

therefore
\[ S_1 = \Phi_T(1) = 1 \]

Three cases may be distinguished:

- If \( M_{X_1} > M_{X_2} \), then \( M_{X_1} > M_{X_2} - 1 \)
- If \( M_{X_1} < M_{X_2} \), then \( M_{X_1} = M_{X_2} - 1 \)
- If \( M_{X_1} = M_{X_2} \), then \( M_{X_1} = M_{X_2} - 1 \)

and consequently
\[ S_1 = \Phi_T(1) = 1 \]

Three cases may be distinguished:

- If \( M_{X_1} > M_{X_2} \), then \( M_{X_1} > M_{X_2} - 1 \)
- If \( M_{X_1} < M_{X_2} \), then \( M_{X_1} = M_{X_2} - 1 \)
- If \( M_{X_1} = M_{X_2} \), then \( M_{X_1} = M_{X_2} - 1 \)

and consequently
\[ S_1 = \Phi_T(1) = 1 \]

Three cases may be distinguished:

- If \( M_{X_1} > M_{X_2} \), then \( M_{X_1} > M_{X_2} - 1 \)
- If \( M_{X_1} < M_{X_2} \), then \( M_{X_1} = M_{X_2} - 1 \)
- If \( M_{X_1} = M_{X_2} \), then \( M_{X_1} = M_{X_2} - 1 \)

and consequently
\[ S_1 = \Phi_T(1) = 1 \]

Three cases may be distinguished:

- If \( M_{X_1} > M_{X_2} \), then \( M_{X_1} > M_{X_2} - 1 \)
- If \( M_{X_1} < M_{X_2} \), then \( M_{X_1} = M_{X_2} - 1 \)
- If \( M_{X_1} = M_{X_2} \), then \( M_{X_1} = M_{X_2} - 1 \)

and consequently
\[ S_1 = \Phi_T(1) = 1 \]

Three cases may be distinguished:

- If \( M_{X_1} > M_{X_2} \), then \( M_{X_1} > M_{X_2} - 1 \)
- If \( M_{X_1} < M_{X_2} \), then \( M_{X_1} = M_{X_2} - 1 \)
- If \( M_{X_1} = M_{X_2} \), then \( M_{X_1} = M_{X_2} - 1 \)

and consequently
\[ S_1 = \Phi_T(1) = 1 \]

Three cases may be distinguished:

- If \( M_{X_1} > M_{X_2} \), then \( M_{X_1} > M_{X_2} - 1 \)
- If \( M_{X_1} < M_{X_2} \), then \( M_{X_1} = M_{X_2} - 1 \)
- If \( M_{X_1} = M_{X_2} \), then \( M_{X_1} = M_{X_2} - 1 \)

and consequently
\[ S_1 = \Phi_T(1) = 1 \]
and consequently
\[ S_2 = \Phi_T(-1) = 0 \]
To summarize
\[ S_2 = \begin{cases} 
1, & \text{if } (E_{X_1} > E_{X_i}) \quad \text{or } (E_{X_2} = E_{X_i} \text{ and } \hat{M}_{X_2} > \hat{M}_{X_i}) \text{,} \\
0, & \text{if } (E_{X_2} < E_{X_i}) \quad \text{or } (E_{X_2} = E_{X_i} \text{ and } \hat{M}_{X_2} \leq \hat{M}_{X_i}) 
\end{cases} \]

Taking into account Definition 4 and Proposition 2, we have
\[ S_2 = \begin{cases} 
1, & \text{if } |X_q| > |X_i| \\
0, & \text{if } |X_q| \leq |X_i| 
\end{cases} \]

According to Definition 1, we may write
\[ \text{comp}([X_i], |X_q|) = S_2 \quad \text{if } i < q. \]

**APPENDIX II**

**Proof of Proposition 4**

Four cases can be distinguished.

1. \( S_{X_2} = S_{X_i} = 0 \implies \Phi_T(-S_{X_q} - S_{X_i} + \text{comp}([X_i], |X_q|) - 1) = \Phi_T(\text{comp}([X_i], |X_q|) - 1) = \text{comp}([X_i], |X_q|) \) (cf. Proposition 1), \( \Phi_T(S_{X_2} + S_{X_i} - \text{comp}([X_i], |X_q|) - 2) = \Phi_T(-\text{comp}([X_i], |X_q|) - 2) = 0 \) (as \( -\text{comp}([X_i], |X_q|) - 2 \leq -2 \)), and \( \Phi_T(-S_{X_q} + S_{X_i} - 1) = \Phi_T(-1) = 0 \) and consequently
\[ \Phi_T(-S_{X_q} - S_{X_i} + \text{comp}([X_i], |X_q|) - 1) + \Phi_T(S_{X_q} + S_{X_i} - \text{comp}([X_i], |X_q|) - 2) + \Phi_T(-S_{X_q} + S_{X_i} - 1) = \text{comp}([X_i], |X_q|). \]

As \( X_q \) and \( X_i \) are simultaneously positive, i.e., \( |X_q| = X_q \) and \( |X_i| = X_i \), we have
\[ \Phi_T(-S_{X_q} - S_{X_i} + \text{comp}([X_i], |X_q|) - 1) + \Phi_T(S_{X_q} + S_{X_i} - \text{comp}([X_i], |X_q|) - 2) + \Phi_T(-S_{X_q} + S_{X_i} - 1) = \text{comp}([X_i], |X_q|). \]

2. \( S_{X_2} = S_{X_i} = 1 \implies \Phi_T(-S_{X_q} - S_{X_i} + \text{comp}([X_i], |X_q|) - 1) = \Phi_T(\text{comp}([X_i], |X_q|) - 3) = 0 \) (as \( \text{comp}([X_i], |X_q|) - 3 \leq -2 \)). \( \Phi_T(S_{X_q} + S_{X_i} - \text{comp}([X_i], |X_q|) - 2) = \Phi_T(-\text{comp}([X_i], |X_q|)) \) and \( \Phi_T(-S_{X_q} + S_{X_i} - 1) = \Phi_T(-1) = 0 \), and consequently (see equation at the bottom of the page).

For \( i < q \)
\[ 1 - \text{comp}([X_i], |X_q|) = \begin{cases} 
1, & \text{if } |X_q| < |X_i| \\
0, & \text{if } |X_q| \geq |X_i| 
\end{cases} \]

\[ = \begin{cases} 
1, & \text{if } |X_i| > |X_q| \\
0, & \text{if } |X_i| \leq |X_q| 
\end{cases} \]

\[ = \text{comp}(X_i, X_q) \] (cf. Definition 1 and (15))

and for \( i > q \)
\[ 1 - \text{comp}([X_i], |X_q|) = \begin{cases} 
1, & \text{if } |X_i| \leq |X_q| \\
0, & \text{if } |X_i| > |X_q| 
\end{cases} \]

\[ = \begin{cases} 
1, & \text{if } |X_i| \geq |X_q| \\
0, & \text{if } |X_i| < |X_q| 
\end{cases} \]

\[ = \text{comp}(X_i, X_q) \] (cf. Definition 1 and (16))

which results in
\[ \Phi_T(-S_{X_q} - S_{X_i} + \text{comp}([X_i], |X_q|) - 1) + \Phi_T(S_{X_q} + S_{X_i} - \text{comp}([X_i], |X_q|) - 2) + \Phi_T(-S_{X_q} + S_{X_i} - 1) = \text{comp}(X_i, X_q). \]

3. \( S_{X_q} = 1 \) and \( S_{X_i} = 0 \) (i.e., \( X_q < X_i \) (cf. Definition 5)) \( \implies \Phi_T(-S_{X_q} - S_{X_i} + \text{comp}([X_i], |X_q|) - 1) + \Phi_T(S_{X_q} + S_{X_i} - \text{comp}([X_i], |X_q|) - 2) + \Phi_T(-S_{X_q} + S_{X_i} - 1) = 0 \) (as \( -\text{comp}([X_i], |X_q|) - 1 \leq -1 \)) and \( \Phi_T(-S_{X_q} + S_{X_i} - 1) = \Phi_T(-2) = 0 \) and consequently
\[ \Phi_T(-S_{X_q} - S_{X_i} + \text{comp}([X_i], |X_q|) - 1) + \Phi_T(S_{X_q} + S_{X_i} - \text{comp}([X_i], |X_q|) - 2) + \Phi_T(-S_{X_q} + S_{X_i} - 1) = 0 \] (as \( -\text{comp}([X_i], |X_q|) - 1 \leq -1 \)) and \( \Phi_T(-S_{X_q} + S_{X_i} - 1) = \Phi_T(0) = 1 \) and consequently
\[ \Phi_T(-S_{X_q} - S_{X_i} + \text{comp}([X_i], |X_q|) - 1) + \Phi_T(S_{X_q} + S_{X_i} - \text{comp}([X_i], |X_q|) - 2) + \Phi_T(-S_{X_q} + S_{X_i} - 1) = 1 = \text{comp}(X_i, X_q) \] (cf. Definitions 1 and 5).

4. \( S_{X_q} = 0 \) and \( S_{X_i} = 1 \) (i.e., \( X_q > X_i \) (cf. Definition 5)) \( \implies \Phi_T(-S_{X_q} - S_{X_i} + \text{comp}([X_i], |X_q|) - 1) + \Phi_T(S_{X_q} + S_{X_i} - \text{comp}([X_i], |X_q|) - 2) + \Phi_T(-S_{X_q} + S_{X_i} - 1) = 0 \) (as \( -\text{comp}([X_i], |X_q|) - 1 \leq -1 \)) and \( \Phi_T(-S_{X_q} + S_{X_i} - 1) = \Phi_T(0) = 1 \) and consequently
\[ \Phi_T(-S_{X_q} - S_{X_i} + \text{comp}([X_i], |X_q|) - 1) + \Phi_T(S_{X_q} + S_{X_i} - \text{comp}([X_i], |X_q|) - 2) + \Phi_T(-S_{X_q} + S_{X_i} - 1) = 1 = \text{comp}(X_i, X_q) \] (cf. Definitions 1 and 5).
REFERENCES


Mohammed Mestari received the M.A. degree from ENSET, Mohammedia, Morocco, in 1991 and the Ph.D. degree in applied mathematics and the Ph.D. degree in artificial intelligence from Hassan II University, Faculty of Science Ben M’Sik Casablanca, Casablanca, in 1997 and 2000, respectively.

He is currently an Assistant Professor of Applied Mathematics at ENSET, Mohammedia, and a Member of the Fundamental Research Unit (URF) of mathematics applied to engineering sciences. His research interests include neural networks for signal processing, neural networks hardware implementation, high-speed and/or low-power techniques and systems for neural networks, and theoretical issues directly related to hardware implementation.