Computing with Nonmonotone Multivalued Neurons

Zoran Obradović [1]

School of Electrical Engineering and Computer Science
Washington State University, Pullman WA 99164-2752, USA

Abstract

Although most of the neural network studies use analog neurons with continuous monotone increasing transfer functions, the reasons for using them are not very well founded. The objective of this paper is to study computational ability of more general neural networks whose transfer functions are not necessarily monotone. A nonmonotone multivalued neural network is proposed as a model for reasoning about certain aspects of the behavior of limited precision analog neural networks with arbitrary continuous transfer functions. The nonmonotone multivalued model is compared to previously studied monotone multivalued and to nonmonotone binary neural networks and it is shown that the models are essentially equivalent. However, the savings in time and hardware arising from using a nonmonotone network rather than monotone can be quite significant as demonstrated on the example of computing symmetric functions and of summing two natural numbers.

[1]Also associated with the Mathematical Institute, Serbian Academy of Sciences and Arts, Belgrade, Yugoslavia. Internet address zoran@eecs.wsu.edu. Partially supported by the NSF research grant NSF-IRI-9308523.

1. Introduction

A *neural network* is a highly parallel nonlinear system consisting of a number of computing units called *neurons*. The interconnection pattern among neurons is a directed weighted graph. A neural network is called *circuit* if the interconnection graph is acyclic and the neurons are organized in *layers* with connections only among neurons in adjacent layers, all starting in a layer that is closer to the input layer and ending in the layer that is closer to the output layer.

In the first order neural network models, a neuron with n incoming connections weighted as $w_1, \ldots, w_n$ computes a function of the form $f(w_1, \ldots, w_n) : \mathbf{R}^n \to S$, where $S \subset \mathbf{R}$, $w_i \in \mathbf{R}$ for $1 \leq i \leq n$, and

$$f(w_1, \ldots, w_n)(x_1, \ldots, x_n) = g(\sum_{i=1}^{n} w_i x_i)$$

for some *transfer function* $g : \mathbf{R} \to S$.

The neural network literature considers mostly neurons with monotone transfer functions. In the *binary model*, each neuron has a two-valued range $S = \{0, 1\}$. If a neuron's threshold is t, then its transfer function g is a *linear threshold function*, defined by $g(x) = 1$ iff $x \geq t$. The binary model is used in the classic perceptron learning algorithm and its more recent extensions [2, 8]. It is one of the computationally best understood neural network models as it is closely related to Boolean threshold circuits theory [1]. However, with the development of the well-known back-propagation learning algorithm for analog neural networks [9], study of such analog models becomes an important research direction. Analog networks appear more useful

in practice, although their computational abilities are still not sufficiently understood. In the *analog model*, $S = [0, 1]$ and g is a continuous increasing function.

For practical purposes, the restriction of limited precision on analog neural networks appears to be a reasonable assumption (provided that the precision is not too small). It was shown earlier that if neuron transfer functions are monotone increasing, then an analog neural network of limited precision can be represented as a *homogeneous k-ary neural network* [4]. In such homogeneous k-ary networks, neurons compute functions of the form $f(w_1, \ldots, w_n) : \mathbf{Z}_k^n \rightarrow \mathbf{Z}_k$, (where $\mathbf{Z}_k = \{0, 1, \ldots, k-1\}$) with $w_i \in \mathbf{Z}$ for $1 \leq i \leq n$, and

$$f(w_1, \ldots, w_n)(x_1, \ldots, x_n) = g(\sum_{i=1}^{n} w_i x_i)$$

for $g(t_1, t_2, \ldots, t_{k-1}) : \mathbf{R} \rightarrow \mathbf{Z}_k$ being defined as $g(x) = i$ iff $t_i \leq x < t_{i+1}$, where thresholds $t_i \in \mathbf{R}$ are monotone increasing and $t_0 = -\infty$, $t_k = +\infty$. Clearly, by analyzing homogeneous k-ary neural networks, we are actually reasoning about certain aspects of behavior of monotone increasing analog neural networks of precision limited to $\log_2 k$ bits [4, 5, 6].

In the homogeneous k-ary neural network model each neuron can have its own set of thresholds. This paper considers an extension to a heterogeneous k-ary neural network model where, in addition, each neuron can have its own set of k monotone increasing output values. Formally, in a *heterogeneous k-ary neural network* each neuron computes a function $f(w_1, \ldots, w_n) : \mathbf{R}^n \rightarrow \mathbf{Z}_k$ for $w_i \in \mathbf{R}$, $1 \leq i \leq n$, where

$$f(w_1, \ldots, w_n)(x_1, \ldots, x_n) = g(t_1, \ldots, t_{k-1}, c_0, \ldots, c_{k-1})(\sum_{i=1}^{n} w_i x_i)$$

and, for some monotone increasing $t_i \in \mathbf{R}$, $1 \leq i < k$, and some monotone nondecreasing $c_i \in \mathbf{R}$, $0 \leq i < k-1$, each neuron's transfer function g is a *generalized multilinear threshold function* $g(t_1, \ldots, t_{k-1}, c_0, \ldots, c_{k-1}) : \mathbf{R} \rightarrow \mathbf{R}$ defined by

$$g(x) = \begin{cases} c_0 & \text{if } x < t_1 \\ c_i & \text{if } t_i \leq x < t_{i+1} \text{ for } 1 \leq i \leq k-2 \\ c_{k-1} & \text{if } x \geq t_{k-1}. \end{cases}$$

Although most of the neural network studies consider only monotone nondecreasing node transfer functions, the reasons for using them are not very well founded. In fact, it is known that some nonmonotone neural networks are quite promising both as computational and learning systems. For example, in the domain of control theory, good practical results are obtained using a nonmonotone neural network model called *radial basis functions* [3]. In this simple nonmonotone model each neuron's transfer function $g(x)$ consists of two monotone regions. The function $g(x)$ is monotone increasing for $x < c$ and is monotone decreasing for $x > c$, where c is a predetermined constant.

A very powerful nonmonotone binary model, called *alternating multilinear neural network*, is studied by Olafsson and Abu-Mostafa [7]. In this model neurons compute *alternating weighted multilinear threshold functions,* $f(w_1, \ldots, w_n) : \mathbf{R}^n \rightarrow \{0, 1\}$ for $w_i \in \mathbf{R}$, $1 \leq i \leq n$, where

$$f(w_1, \ldots, w_n)(x_1, \ldots, x_n) = g(\sum_{i=1}^{n} w_i x_i)$$

and, for some monotone increasing $t_i \in \mathbf{R}$, $1 \leq i < k$, and $t_0 = -\infty$, $t_k = +\infty$, each

neuron's transfer function g is an *alternating threshold function* $g(t_1, t_2, \ldots, t_{k-1})$: $\mathbf{R} \to \{0, 1\}$ defined by

$$g(x) = \begin{cases} 0 & \text{if } t_{2i} < x < t_{2i+1} \text{ for some } 0 \leq i < k/2 \\ 1 & \text{otherwise.} \end{cases}$$

This paper extends homogeneous and heterogeneous k-ary as well as alternating multilinear neural networks to more general *nonmonotone k-ary neural networks.* Their neurons compute functions of the form $f(w_1, \ldots, w_n) : \mathbf{R}^n \to \mathbf{Z}_k$ for $w_i \in \mathbf{R}$, $1 \leq i \leq n$, where

$$f(w_1, \ldots, w_n)(x_1, \ldots, x_n) = g(t_1, \ldots, t_{k-1}, c_0, \ldots, c_{k-1})(\sum_{i=1}^{n} w_i x_i)$$

and, for some monotone increasing $t_i \in \mathbf{R}$, $1 \leq i < k$, and some $c_0, \ldots, c_{k-1} \in \mathbf{Z}_k$, each neuron's transfer function g is a *nonmonotone multilinear threshold function* $g(t_1, \ldots, t_{k-1}, c_0, \ldots, c_{k-1}) : \mathbf{R} \to \mathbf{R}$ defined by

$$g(x) = \begin{cases} c_0 & \text{if } x < t_1 \\ c_i & \text{if } t_i \leq x < t_{i+1} \text{ for } 1 \leq i \leq k-2 \\ c_{k-1} & \text{if } x \geq t_{k-1}. \end{cases}$$

Nonmonotone k-ary neural networks correspond to limited precision analog neural networks whose transfer function is not necessarily monotone nondecreasing. In this paper we considered the computational and learning ability of such nonmonotone k-ary neural networks, comparing them to previously studied monotone k-ary and to nonmonotone binary models.

The paper is divided into four sections. In Section 2 the relationship between the nonmonotone networks and the homogeneous k-ary networks, as well as the relationship between the nonmonotone networks and the alternating neural networks,

is given. In Section 3 a resource trade-off involving depth, size and weights among various multivalued neural network models is explored.

2. Relationship Between Models

This section considers the resources of running time, size, depth and weight of monotone k-ary and alternating multilinear neural networks needed to simulate given non-monotone k-ary neural networks. A neural network M_2 is $f(t)$-*equivalent* to network M_1 iff for all inputs x, for every computation of M_1 on input x which terminates in time t there is a computation of M_2 on input x which terminates in time $f(t)$ with the same output. If $f(t)$ is the identity function, we simply say that M_2 is *equivalent* to M_1. The *depth* of a neural network is measured as the number of layers, and the *size* as the number of neurons in a network. The input neurons are not counted in either the depth or the size. The *weight* of a neural network is the sum of the absolute values of all the weights in the network's interconnection graph.

It is known that the alternating multilinear neural network model is closely related to the homogeneous k-ary model:

Theorem 1 *For every homogeneous k-ary neural network of size z and weight w there is an equivalent alternating multilinear neural network of size $z \log_2 k$ and weight $(k-1)w \log_2 (k-1)$ which produces the output of the former in binary representation. Also, for every alternating multilinear neural network of size z and weight w, there is a $3t$-equivalent homogeneous k-ary neural network of size $4z$ and weight $w + 4z$.*

Proof: See Obradović and Parberry [4]. □

However, to simulate a nonmonotone k-ary neural network of size z, clearly at least kz homogeneous k-ary or alternating neurons are needed. The following theorem shows that this lower bound is tight and can be achieved by a binary network which is a special case of the homogeneous k-ary and also of the alternating multilinear model.

Theorem 2 *For every nonmonotone k-ary neural network of size z and weight w there is an equivalent binary neural network of size kz and weight $O(ckw)$. Here, $c = \max_{v \in V} \max_{i \in \mathbf{Z}_k} |c_i^v|$, where c_i^v is the i^{th} output value of a nonmonotone neuron v and V is the set of all neurons in the network.*

Proof: A nonmonotone k-ary neuron v with output values $c_0^v, \ldots, c_{k-1}^v$ and thresholds $t_1^v, \ldots, t_{k-1}^v$ is replaced by a single layer circuit F^v consisting of k binary neurons $v^0, v^1, \ldots, v^{k-1}$. A binary neuron v^i, $0 \leq i \leq k-1$, has threshold t_i^v, where t_0^v is defined as $t_0^v = \infty$. The incoming connections of a binary neuron v^i and corresponding weights are identical to incoming connections and weights of a nonmonotone neuron v. If w_{vu} is a connection weight from neuron v to neuron u in a nonmonotone k-ary circuit, then in the corresponding binary circuit the weight from neuron v^i, $0 \leq i \leq k-1$, to neuron u is set to $(c_i^v - c_{i-1}^v)w_{vu}$, where c_{-1}^v is defined to be zero. For a nonmonotone k-ary neuron with thresholds $t_1^v, \ldots, t_{k-1}^v$ shown in Figure 1 (a), the corresponding equivalent binary circuit F^v is constructed as shown in Figure 1 (b).

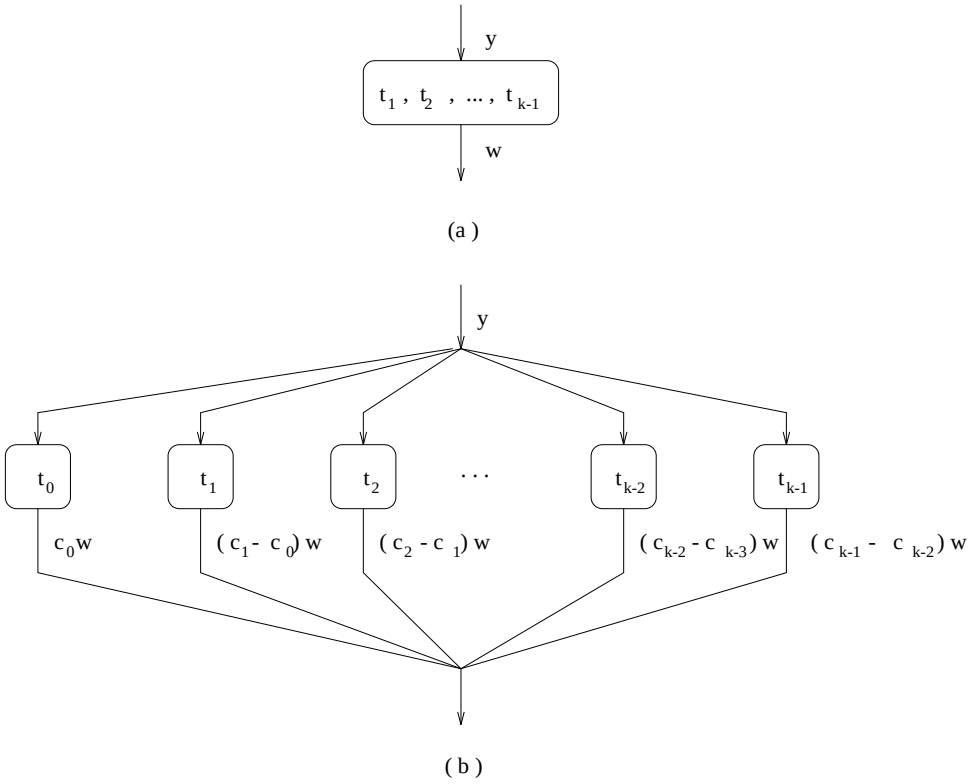

Figure 1: A Nonmonotone Neuron (a) vs. Corresponding Binary Network (b)

Let us assume that the weighted input sum y for a nonmonotone k-ary neuron v satisfies $t_j \leq y < t_{j+1}$. Then the weighted input from neuron v to neuron u is $w_{vu}c_j^v$. By construction of the binary subcircuit F^v, it is clear that the weighted input from the binary subcircuit F^v to each gate in the binary subcircuit F^u corresponding to the nonmonotone k-ary neuron u is

$$w_{vu} \sum_{i=0}^{j} (c_i^v - c_{i-1}^v)$$

which also sums to $w_{vu}c_j^v$ proving the equivalence claim from the theorem.

The key to the weight bound is the observation that

$$\sum_{i=0}^{k-1} |w_{vu}(c_i^v - c_{i-1}^v)| \leq 2ck|w_{vu}|.$$

□

In a *multicase neural network* each neuron computes a function of the form $f(w_1, \ldots, w_n) : \mathbf{R}^n \to \mathbf{Z}_k$ for $w_i \in \mathbf{R}$, $1 \leq i \leq n$, where

$$f(w_1, \ldots, w_n)(x_1, \ldots, x_n) = g(t_1, \ldots, t_{k-1}, c_0, \ldots, c_{k-1})(\sum_{i=1}^{n} w_i x_i)$$

and, for some monotone increasing thresholds $t_i \in \mathbf{R}$, $1 \leq i < k$, and some output values $c_1, \ldots, c_{k-1} \in \mathbf{Z}_k$, the neuron's transfer function $g(t_1, \ldots, t_{k-1}, c_0, \ldots, c_{k-1}) : \mathbf{R} \to \mathbf{R}$ is defined by

$$g(x) = \begin{cases} c_i & \text{if } x = t_i \text{ for } 1 \leq i \leq k-1 \\ 0 & \text{otherwise.} \end{cases}$$

A multicase neural network is called a *homogeneous k-case neural network* if each of its multicase neurons can have up to $k-1$ thresholds with the output values

$c_i = i$, $1 \leq i \leq k-1$. Homogeneous k-case neural networks are closely related to homogeneous k-ary neural networks as shown by the following theorem.

Theorem 3 *For every homogeneous k-case neural network of size z and weight w there is a $3t$-equivalent homogeneous k-ary neural network of size $5z$ and weight $O(w+kz)$.*

Proof: A k-case neuron v with thresholds $t_1, \ldots, t_{k-1}$ and outputs $c_i^v = i$, $1 \leq i \leq k-1$, is replaced by a k-ary circuit F^v of depth 3 and size 5. The first layer of F^v has two homogeneous k-ary neurons v' and v'', the first with thresholds $t_1, \ldots, t_{k-1}$, and the second with thresholds $-t_{k-1}, \ldots, -t_1$. The weighted inputs to v' are same as for v in the original circuit. The weights of the inputs to v'' are negated. That means that if in the original circuit the weight from input x_i to neuron v is y_i, then in F^v the weight from input x_i to neuron v'' is $-y_i$. The second layer has two neurons u' and u''. Here, u' is a homogeneous k-ary neuron with thresholds $1, 2, \ldots, k-1$ and with unit weight connection from neuron v'. Neuron u'' has inputs from v' and v'', both with weight -1. It checks whether its weighted input sum x is $x \geq -k+1$. The output layer of F^v has a single homogeneous k-ary neuron with thresholds $1, 2, \ldots, k-1$. Its first input with weight 1 is from u', and the second input with weight $-k$ is from u''. It is easy to see that such a homogeneous k-ary circuit satisfies the claim of the theorem. □

The following corollary gives a general relation between multicase neural networks

and nonmonotone neural networks.

Corollary 4 *For every k-case neural network of size z and weight w there is an $3t$-equivalent nonmonotone k-ary neural network of size $5z$ and weight $O(w + kz)$.*

Proof: A k-case neuron v with thresholds $t_1, \ldots, t_{k-1}$ and outputs $c_1^v, \ldots, c_{k-1}^v$ is replaced by a nonmonotone k-ary circuit F^v of depth 3 and size 5. The only difference between F^v from this corollary and Theorem 3 is the output values of a neuron in the third layer are $0, c_1^v, \ldots, c_{k-1}^v$ instead of previous outputs $0, 1, \ldots, k-1$. □

3. Resource Trade-offs

Some upper-bound results for the depth, size and weight of homogeneous k-ary neural networks with all weights drawn from $\{\pm 1\}$ are explored in [4]. Here we demonstrate that the improvements of these upper bounds are possible using heterogeneous k-ary and nonmonotone k-ary neural networks.

Let $x = x_1 \ldots x_n$ and $y = y_1 \ldots y_n$, where $x_i, y_i \in \mathbf{Z}_k$ for $1 \leq i \leq n$, be two natural k-ary numbers of *size* n. In order to compute sum $z = x + y$, where $z = z_1 \ldots z_{n+1}$, let us first consider the problem of computing the *carry* $c_1 \ldots c_{n+1}$ of x and y. Here, c_i is defined to be 1 if there is a carry into the i^{th} position of the result, that is, into z_i.

Theorem 5 *The carry of two natural k-ary numbers of size n can be computed by a homogeneous k-ary neural circuit with size $O(n^2)$, depth 3 and all weights drawn from $\{\pm 1\}$.*

Proof: See Obradović and Parberry [4]. □

However, the carry can be computed even in a single layer using linear number of neurons if unit weights are not our primary concern.

Theorem 6 *The carry of two natural k-ary numbers of size n can be computed by a homogeneous k-ary neural circuit of size n and depth* 1.

Proof: Carry c_i into the i^{th} position of $x + y$ is 1 if the sum of two k-ary numbers $x^{(i)} = x_i \ldots x_n$ and $y^{(i)} = y_i \ldots y_n$ is at least k^{n-i}. Obviously, c_{n+1} is always zero. Each c_i, for $1 \leq i \leq n$, can be computed by a single k-ary neuron v_i with $2(n-i+1)$ inputs from x_j and y_j for $i \leq j \leq n$. The neuron v_i has its first threshold set to k^{n-i} and all other thresholds set to very large values that are never exceeded by the weighted input sum. Finally, the weights w_{ji} from inputs x_j and y_j to the neuron v_i are set to k^{n-j} for $i \leq j \leq n$. □

The consequence of the Theorem 6 is a smaller depth and a small size neural circuit for computing the sum of two k-ary numbers.

Corollary 7 *The sum of two k-ary integers of size n can be computed by a homogeneous k-ary neural circuit of depth 5 and size $O(n^2)$ with all weights drawn from*

$\{\pm 1\}$. *If the weights are not restricted to unit values, the complexity of a homogeneous k-ary neural circuit to compute this sum can be reduced to depth* 3 *and size* $3n+2$.

Proof: For a unit weight solution, first compute the carry of x and y in quadratic size and depth 3 using Theorem 5. Then each z_i can be computed from x_{i-1}, y_{i-1}, and c_i in constant size and depth 2. A reduced circuit is obtained by using Theorem 6 instead of Theorem 5. □

However, the complexity of the neural circuit can be further reduced by using nonmonotone $(2k-1)$-ary neurons.

Corollary 8 *The sum of two k-ary integers of size n can be computed by a nonmonotone $(2k-1)$-ary neural circuit of size $n+1$ and depth* 1.

Proof: Each z_i can be computed from $x_{i-1}, \ldots, x_n$ and $y_{i-1}, \ldots, y_n$ using a single $(2k-1)$-ary nonmonotone neuron v^i whose weights $w_{j,i}$, $i < j \leq n$ from inputs x_j and y_j are set to k^{n-j}. The $2k-1$ thresholds of neuron v^i are $t_1^i = k^{n+1-i}$, $t_2^i = 2k^{n+1-i}$, $t_3^i = 3k^{n+1-i}, \ldots, t_{2k-1}^i = (2k-1)k^{n+1-i}$ and the output values c_l^i, $0 \leq l \leq 2k-1$ are defined as

$$c_l = \begin{cases} l & \text{if } l < k, \\ l-k & \text{otherwise.} \end{cases}$$

□

Consider now a trade-off between the complexity of neurons and the size and depth of neural networks on a problem of computing an arbitrary k-ary symmetric function

of n inputs by neural networks. There has been much interest in this problem as it is believed that natural neural networks are very efficient in computing symmetric functions, while computing those functions on standard parallel processing systems is computationally expensive. Formally, a function $f : \mathbf{Z}_k^n \to \mathbf{Z}_k$ is called *symmetric* if its output remains the same no matter how the input is permuted. That is, for all one-to-one and onto $\Pi : \{1, \ldots, n\} \to \{1, \ldots, n\}$, and all $x_1, \ldots, x_n \in \mathbf{Z}_k$,

$$f(x_1, \ldots, x_n) = f(x_{\Pi(1)}, \ldots, x_{\Pi(n)}).$$

Theorem 9 *Any k-ary symmetric function on n inputs can be computed by a homogeneous k-ary neural circuit of depth 6 and size*

$$\frac{(n+1)^{k-1}}{(k-2)!} + 6kn + 3k + 1$$

with all weights drawn from $\{\pm 1\}$.

Proof: See Obradović and Parberry [4]. □

If unit weights are not a primary concern, the same symmetric function can be computed with a smaller size and depth.

Theorem 10 *Any k-ary symmetric function on n inputs can be computed by a homogeneous k-ary neural circuit of depth 5, weight $O(n^{2k})$ and size*

$$\frac{(n+1)^{k-1}}{(k-1)!} + 3kn + 1.$$

Proof: Let

$$S = \{(y_0, \ldots, y_{k-1}) \in \mathbf{N}^k | \sum_{i=0}^{k-1} y_i = n\}.$$

and

$$S' = \{\sum_{j=0}^{k-1} n^{y_j} : (y_0, \ldots, y_{k-1}) \in S\}.$$

The function $f : \mathbf{Z}_k^n \rightarrow \mathbf{Z}_k$ is symmetric, and so for each input $x = (x_1, \ldots, x_n) \in \mathbf{Z}_k^n$ for which

$$\sum_{i=1}^{n} n^{x_i} = s$$

follows

$$f(x) = f_s$$

where $f_s \in \mathbf{Z}_k$ is a constant.

The unit weight neural circuit of Theorem 9 computes the number of 0's, 1's,...,$(k-1)$'s in the input and its output is based on the outcome of those k tests. Instead, here the neural circuit is reduced by computing

$$\sum_{i=1}^{n} n^{x_i} = s$$

and checking if this sum is an elements of set $S' = \{s_1, \ldots, s_m\}$. If the sum s is equal to $s_i \in S'$, the output is f_{s_i}. Consequently, the first layer of the circuit consists of nk subcircuit T_j^i, $1 \leq i \leq n$ and $0 \leq j \leq k-1$, of depth 2 and size 3, where T_j^i tests whether $x_i = j$. The next layer consists of m subcircuit S^h, $1 \leq h \leq m$, of depth 2 and size 3, where S^h tests whether its weighted input sum is equal to $s_h \in S'$. here, each S^h has inputs from all subcircuits T_j^i $(1 \leq i \leq n$ and $0 \leq j \leq k-1)$, while The

weight from T_j^i to S^h is set to n^j. Finally, the weight from S^h, $1 \leq h \leq m$, to a single homogeneous k-ary neuron in the output layer with thresholds $1, 2, \ldots, k-1$, is set to f_{s_h}. $\square$

A symmetric function can be computed even in a smaller depth and size using a neural circuit with heterogeneous k-ary neurons.

Corollary 11 *Any k-ary symmetric function on n inputs can be computed by a heterogeneous k-ary neural circuit of depth 4, with unit weights and size*

$$\frac{(n+1)^{k-1}}{(k-1)!} + n + 1.$$

Proof: The first layer of the circuit has n heterogeneous k-ary neurons $v_1, \ldots, v_n$. Neuron v_i with thresholds $1, 2, \ldots, k-1$ and output values $c_0 = 0$, and $c_i = n^{i-1}$, for $1 \leq i \leq k-1$ has a unit weight on the input connection from x_i. Subcircuits S^h, $1 \leq h \leq m$, of depth 2 and size 3 are defined similarly as in Theorem 10. The only differences are that the output neuron of S^h is a heterogeneous k-ary neuron with output values 0 and f_{s_h}, and the weight from S^h to the output neuron is 1. It is easy to see that this heterogeneous k-ary neural circuit is equivalent to the homogeneous k-ary neural circuit from Theorem 10. $\square$

Drastic decrease in depth and size is possible if neuron transfer functions are further modified to multicase functions as follows.

Corollary 12 *Any k-ary symmetric function on n inputs can be computed by a multicase neural circuit of depth 2, with unit weights and size* $n + 1$.

Proof: The first layer of the circuit has n multicase neurons $v_1, \ldots, v_n$. Neuron v_i has thresholds $1, 2, \ldots, k-1$, output values $c_0 = 0$, $c_i = n^{i-1}$, for $1 \leq i \leq k-1$ and a unit weighted input from x_i. The outputs of all v_i, $1 \leq i \leq k$ are connected with unit-weight edges to a single multicase neuron in the second layer. Thresholds of that multicase neuron are $s_1, s_2, \ldots, s_m$, where $S' = \{s_1, s_2, \ldots, s_m\}$ is defined as in Theorem 10. The output values of the output neuron are f_{s_i} $(0 \leq i \leq m)$. □

4. Conclusions

In this paper the computational abilities of several multivalued neural network models are explored. Study of these models has practical interest as they are closely related to limited precision analog neural networks with nonmonotone transfer functions. The $f(t)$-equivalence relation between different models has been shown. However, a tradeoff involving resources of depth, size, weight and the complexity of neuron transfer functions, demonstrated on the example of computing symmetric functions and of summing two natural numbers, indicates that significant savings are possible through selection of an appropriate nonmonotone multivalued model. Further research is needed to develop a general theory for identifying the optimal trade-off between complexity of neurons versus size and depth of a neural network designed

using those neurons as its building blocks. Also, further research is needed to develop efficient learning methods for identified nonmonotone multivalued neural networks.

Acknowledgements

I would like to thank Ian Parberry, whose encouragement of my work on this project has been invaluable. Further, I thank Radu Drossu for his constructive comments on a preliminary version of the manuscript. Finally, I am grateful to three anonymous reviewers for their careful reading and fast handling of the manuscript, and most grateful to a reviewer who pointed out a simple way of improving Theorem 2 and Corollary 8.

References

[1] P. Dunne, The Complexity of Boolean Networks, Academic Press, 1988.

[2] S.I. Gallant, Perceptron-Based Learning Algorithms, *IEEE Trans. Neural Networks* **1** (1990), pp. 179-191.

[3] J. Moody, and C. Darken, Learning with Localized Receptive Fields, *Proc. 1988 Conectionist Models Summer School,* (1988), pp. 133-143.

[4] Z. Obradović, and I. Parberry, Computing with Discrete Multivalued Neurons, *J. Comput. System Sci.* **45** (1992), pp. 471-492.

[5] Z. Obradović, and I. Parberry, Learning with Discrete Multivalued Neurons, *J. Comput. System Sci.* **49** (1994), pp. 375-390.

[6] Z. Obradović, and P. Yan, Small Depth Polynomial Size Neural Networks, *Neural Computation,* **2** (1990), pp. 402-404.

[7] S. Olafsson, and Y.S. Abu-Mostafa, The Capacity of Multilevel Threshold Functions, *IEEE Trans. Pattern Analysis and Machine Intelligence,* **10** (1988), pp. 277-281.

[8] F. Rosenblatt, Principles of Neurodynamics, Spartan, 1962.

[9] D.E.Rumelhart, G.E.Hilton and R.J.Williams, Learning Internal Representations by Error Propagation, in Parallel and Distributed Processing, Eds. D.E.Rumelhart and J.L.McClelland, Cambridge, MA, MIT Press, 1986.