NEUCOM 203

Cost-performance evaluation of analog neural networks and high order networks

M. Sipper and Y. Yeshurun

Department of Computer Science, School of Mathematical Sciences, Sackler Faculty of Exact Sciences, Tel Aviv University, 69978, Tel Aviv, Israel

Received 9 April 1992
Revised 17 August 1992

Abstract

High order networks, studied over the past few years, have been shown to improve learning rates, increase storage capacity and reduce the number of layers required in comparison with first order nets. One issue which usually remains in the background, is the relative cost-performance of such nets. In this paper we address this issue in a more general framework, which we define, namely generalized high order networks. We present a cost-performance model and demonstrate its usability by analyzing some well-known first and high order networks. Our aim is to provide a simple, yet illuminating model, which enables the evaluation and analysis of generalized high order networks.

Keywords. Cost-performance analysis; analog neural networks; high order networks.

1. Introduction

Artificial neural networks have been an object of intense, diverse research over the past few years. This interest has been fueled by new theoretical results and by advances in computer technology that make it possible to simulate networks of much higher complexity than was possible before. Moreover, microelectronic technology has reached a stage where large neural networks can be integrated onto a single chip. *Analog* implementations of neural networks were built as early as the 1960s [37, 23], however, they were small scale and used discrete components.

Electronic neural networks rely on strongly simplified models of neurons. Analog neural network implementations [10, 8] use analog components and computations. Neuronal inputs and outputs are voltages and synapses (interconnections) are implemented as resistors with conductances as weight values. The transfer function is generally implemented as an amplifier. All the currents coming from other neurons are summed on the input wire and the output voltage of the neuron is a function of this total current and the transfer function.

Correspondence to: M. Sipper, Department of Computer Science, School of Mathematical Sciences, Sackler Faculty of Exact Sciences, Tel Aviv University, 69978, Tel Aviv, Israel.

Computing sums of products is a key operation performed by the network and a hardware implementation has to focus on doing this efficiently [10]. Very often only modest precision is required so that it is possible to use analog computation for this task. In an analog network, a single resistor can perform a multiplication using Ohm's law, and summing of currents on a wire is provided by Kirchhoff's law.

There are a number of issues which make the use of analog circuits attractive [8]. First, analog circuits are faster than digital implementations in terms of speed-to-amount of hardware ratio. Second, since these analog components require less circuitry, it is possbile to pack more components onto a single VLSI chip. The simplicity of these devices also makes them very attractive for rapid prototyping. Third, since the analog conductance devices and processing elements operate in small voltage swings, they dissipate less power and this reduces the problem of heat transfer. Fourth, some researchers believe that analog circuits provide us with a better understanding of the true analog nature of biological neural networks. Analog circuits also have an edge over digital designs in the density of the connections. A recent review [11] of electronic neural networks in the US and Canada revealed that over 40 different circuits were built during the last two years. A majority of these, over 30 designs, use analog computation to some extent.

High order networks, which replace the linear neuron with a general polynomial one, have been an object of research in the past few years [20, 25, 9, 14, 7, 28]. In the next section we present a generalized model which we shall use in this paper.

2. Generalized high order networks

The use of general types of neural network models has received little attention. Feldman [6] defines an abstract, general computational model, in which each element performs some general function. However, he then treats more conventional models. Recently, Hecht-Nielsen [12] defined a neural network as a general parallel, distributed information processing model. He then goes on to express the opinion that there is no satisfactory answer as to why we do not simply study neural networks as MIMD (Multiple Instruction Multiple Data) parallel machines. In his opinion there is only the empirical evidence that neural nets do produce powerful and potentially useful information processing structures.

The general framework adopted in this paper is one which we term *generalized high order networks*. Such a net consists of neurons computing the following function:

$$y_i = \left\{ w_i + \sum_j f^{(1)}(w_{ij}, x_j) + \sum_{jk} f^{(2)}(w_{ijk}, x_j, x_k) + \ldots \right\}$$

$$o_i = F(y_i) \, ,$$

where

y_i	is the total input to the ith neuron
x_j	is the input from the jth neuron
$w_i, w_{ij}, w_{ijk}\ldots$	represent 0th,1st,2nd,... order weights
$f^{(t)}$	is the t order function
o_i	is the output of the neuron
F	is some arbitrary nonlinearity.

In the prevailing model [20, 25, 9, 14, 7, 28] $f^{(t)}$ is a t-way multiplication and the implementation follows in the lines of Section 1, with one addition. The high order network function includes a multiplication of the inputs (e.g. $x_j x_k$) which is implemented using an analog voltage multiplier (such a component may be constructed using a differential amplifier and some additional resistors [22]).

The computational power of high order networks was conjectured to be superior to that of linearly interconnected nets. However, recently [30] it has been proven that one can simulate all Turing machines by recurrent nets using only first order (i.e. linear) connections. Furthermore, this simulation can be done in linear time.

The issue at hand is therefore not one of *computational* capabilities but of *cost performance*. This question usually remains in the background, and becomes even more acute when dealing with generalized nets. Researchers have used high order networks to attain increased learning rates [9, 21, 29] and increased storage capacity [25, 3] in relation to first order nets. Also in many cases it is easier to train a high order network than a multi-layer net since training the hidden layers is more difficult [21].

It is evident that a method for analyzing generalized high order networks is required. Our goal in this paper is to present a cost-performance model which enables such an analysis. Using our model it is possible to evaluate networks and derive exact expressions which enable the comparison of their relative strengths. In Section 3 we present a cost measure and in Section 4 a performance measure. We then combine both and define the cost-performance measure. In Section 5 we demonstrate the use of our model by analyzing several networks and finally we conclude in Section 6.

3. The cost measure

In measuring the cost of generalized high order networks we are interested in a *simple* and *practical* model. Following these guidelines we provide a cost measure which is essentially the number of basic components of the network. This number is very simple to compute. Since high order networks, as also first order nets, consist of a large number of uniform elements we need only consider one and then multiply by the size of the net. The cost measure is also practical, since the number of components is directly related to the size (area) of the chip, the main consideration in VLSI [17].

The basic components, denoted *the base set*, are:

$$\{R \text{ (resistors)}, C \text{ (capacitors)}, D \text{ (diodes)}, T \text{ (transistors)}, L \text{ (inductors)}\}.$$

The most basic set of components in analog electronics is that of resistors, capacitors and inductors [4, 27]. However, as we are interested in a practical framework we add diodes and transistors. These are of such common, basic use that we include them in the base set. The inductor is included solely for completeness purposes although it is rarely used due to its high cost in VLSI. We do not include wires, since we are dealing in high order models wich are essentially synaptic in nature and thus a wire generally includes a resistor or a capacitor. In cases where this is not true a zero-valued resistor is assumed. A related issue which is not taken into account in our model is that of wire *lengths* (e.g. [17] discusses various VLSI efficiency measures including: wire area – sum of the lengths of the wires and maximum edge length – length of the longest wire). Our reason for this decision is that measures involving wire lengths are much more difficult to compute, especially in a theoretical setting. In the tradeoff between simplicity and precision we opt for simplicity for reasons which are explained below (see also Section 4).

The cost measure, CO, consists of the weighted sum of the number of basic components and is defined as:

$$CO = a_1 R + a_2 C + a_3 D + a_4 T + a_5 L \ ,$$

where

R, C, D, T, L denote the number of the respective components of that type
$a_1, ..., a_5$ are the relative costs of the basic components.

This measure is computed for various networks, providing insight as to their relative cost. Such an approach has also been taken in digital implementations of neural nets. For example Lauwereins and Bruck [16] use a cost measure of the number of gates in a circuit, each multiplied with its number of inputs. Again this is directly related to chip size.

Different values of a_i provide for relative component costs. A related issue is that of differences among components of the same type. For example, the backpropagation model [26] requires at least an 8-bit weight representation for a large problem of practical interest [8, 10]. Thus, in general, analog networks are more appropriate for models requiring moderate or low precision. We therefore consider only one type of each component (it is straightforward to enhance the base set although this is rarely needed since higher resolution components can be built of lower resolution ones [8]). In this paper we assume $a_i = 1, i = \{1..5\}$.

There exist various VLSI models which basically measure the area of a circuit and its operation time (see for example AT^2 models, [35]), however, our motivation is different. Whereas in the general arena of analog VLSI, differences between chips may be subtle, thus requiring a precise model of comparison, the differences between generalized high order networks are usually much more evident. Thus our cost measure has been defined with the intention of capturing the essence of these larger differences and as such, we were able to retain simplicity.

4. The performance measure

The two most prominent gains cited in connection with high order networks are increased learning rates and increased storage capacity. Thus, we define two performance measures:

TI	The operation time of a network.
STO	The storage capacity of a network: The number of patterns it is able to store without producing spurious output (note that in many cases this is given by empirical findings and not by rigorous analysis).

The question of time, while theoretically uninteresting in analog computation, is nonetheless important in our practical framework. We therefore assume the existence of discrete time steps and define a *basic time unit* as the time required for a signal to propagate through a base set component (we assume that signal propagation time is equal for all components). The total operaton time of a network is the number of basic time units that elapse from the first input signal until the last output signal. We need not fix on the notion of a time unit to within closer than a constant factor, because our results about the time required for solution will not be more precise than that.

We distinguish between two operational phases of the network: The *Learn* phase involves the setting of weights while the *Run* phase involves computing outputs of novel inputs. (These two stages are not necessarily separate and a network may operate in a single Run & Learn phase in which weights are updated continuously). We thus define two sub-measures of time:

- TI_L The number of basic time units required to learn *one* input vector.
- TI_R The number of basic time units required to run *one* input vector.

Note that we are dealing in *averaged* units in the sense that time is measured per one vector. The time measure TI is defined as:

$$TI = TI_L + TI_R \,.$$

As in Section 3 here too we have made a reasonable, but potentially unrealistic assumption about the delays that wires introduce (these assumptions are in accordance with those made in VLSI computational models discussed in [35]). We have assumed that one time unit is sufficient for signals to propagate down wires, as well as to switch transistors. In reality, as wires get longer, the time of propagation cannot be regarded as constant. We have adopted the constant propagation time model for two reasons (see [35]). First, there seem to be few opportunities, in VLSI in general, to prove stronger lower bounds on time by making stronger assumptions on wire delays. Second, in practice, wire delays do not seem to dominate switching time, except for delays on a few long wires. In those cases, we can often reduce these wire delays by driving signals down them with low resistance pullups.

First and high order networks are essentially parallel models, whose parallelism manifests itself in their layered mode of operation. All elements of a layer are assumed to operate in parallel, i.e. compute their output in $O(1)$ time. Thus, feedforward networks operate in $O(1)$ time while recurrent nets usually operate in non-constant times (see examples in Section 5).

The performance of a network improves with increases in storage capacity (STO) and decreases in operation time (TI). Thus we define the network's *performance* measure, PER:

$$PER = \frac{STO}{TI} \,.$$

The total cost-performance measure, COP, of a generalized high order network is defined as:

$$COP = \frac{PER}{CO} \,.$$

5. Examples

5.1 Comparing a first order network and a second order network

Some common problems found in the neural network literature are those of: symmetry (classifying input vectors as to whether or not they are symmetric about their center), parity (classifying binary vectors as to whether they contain an odd or an even number of 1s) and contiguity (detecting various contiguous patterns in a given input vector). These may be solved using two types of approaches. The first approach involves the use of first order networks, where one of the prevailing models is that of backpropagation. This model employs a first order, three layer neural network, composed of an n neuron input layer, an l neuron hidden layer, and an m neuron output layer (see [26, 19, 20]). The second approach involves the use of a second order network. This model employs a second order, two layer network composed of an n neuron input layer and an m neuron output layer. The nets are fully connected and feedforward (see *Fig. 1*). Both models have successfully solved the above problems. Thus, as noted in Section 2, the issue at hand is not one of computational capabilities, but of cost-performance.

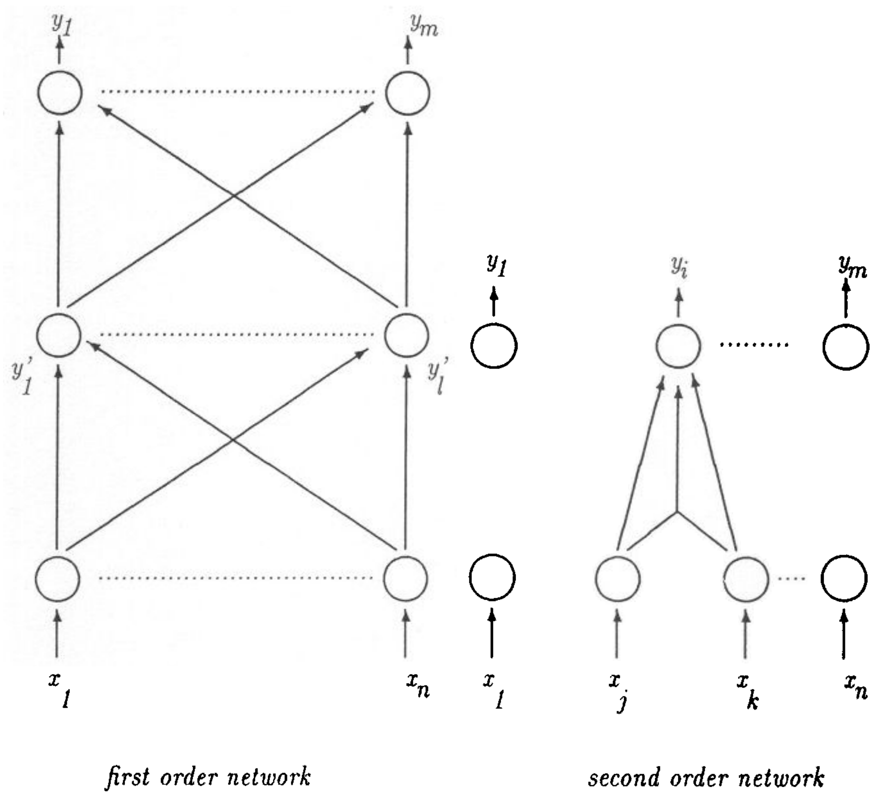

Fig. 1. A schema of a first and second order network.

The input layer neurons and the transfer functions are ignored, since these are equal in both networks. The first order network consists of l neurons of n resistors plus m neurons of l resistors. Thus, the cost of the first order net, denoted CO_1, is $l(n + m)$.

The second order network includes m second order output neurons. Each output neuron consists of n basic components (resistors) of the first order term and $n^2(1 + CO_M)$ basic components of the second order term, where CO_M is the cost of the multiplier. Therefore, CO_2, the cost of the second order network, is $m\big(n + n^2(1 + CO_M)\big)$. Thus, $CO_1 = O\big(l(n+m)\big)$ and $CO_2 = O(mn^2)$.

Using the two cost measures it is possible to compute the number of hidden layer neurons, for which the second order net is less costly, i.e. $CO_2 < CO_1$:

$$m\big(n + n^2(1 + CO_M)\big) < l(n+m)$$

$$l > \frac{m\big(n + n^2(1 + CO_M)\big)}{n+m} .$$

This result demonstrates a common tradeoff between nets of different orders and layers. While a second order network is usually composed of fewer layers (indeed this has commonly been cited as one of their advantages) they are not necessarily less *costly*. The above result states the precise break-even point, in terms of cost alone.

After obtaining the cost measure we continue with the cost-performance analysis. Both networks use one shot learning and are feedforward, hence $TI_1 = TI_2 = O(1)$, where TI_1 and TI_2 are the operation times of the networks. The storage capacities of the networks are usually obtained through simulations. However, it is possible, using our model, to derive an analytical result which states explicitly the capacity for which the cost-performance of the second order network is improved.

Denote the network capacities by STO_1 and STO_2 for the first and second order networks, respectively. The networks cost-performance measures are:

$$COP_1 = O\Big(\frac{STO_1}{l(n+m)}\Big)$$

$$COP_2 = O\Big(\frac{STO_2}{mn^2}\Big) .$$

Our goal is: $COP_2 > COP_1$. Thus:

$$O\Big(\frac{STO_2}{mn^2}\Big) > O\Big(\frac{STO_1}{l(n+m)}\Big)$$

$$O\Big(\frac{STO_2}{STO_1}\Big) > O\Big(\frac{mn^2}{l(n+m)}\Big) .$$

Since both networks operate in $O(1)$ time, and the cost of the second order net is higher, the gain lies in the improved storage capacity. Our model has provided us with an analytical result which acts as a guideline to measuring the successful implementation of a generalized high order network.

As an example consider $m \approx n \approx l = O(m)$. In this case:

$$O\Big(\frac{STO_2}{STO_1}\Big) > O(m) .$$

This simplified example demonstrates the use of our model. The result states explicitly the break-even point at which a second order network becomes cost-efficient.

Note that there is an inherent cost involved in the second order network, which grows as n^2. Thus, efforts to decrease the cost of this net should concentrate on the n^2 term. This is no simple task since it is evident that even if we succeed in reducing the cost of the multiplier, CO_M (see for example [24] for a multiplier of $CO_M = 3$), we still have $O(n^2)$ components, due to the weights. A possible approach is that of using a sparse network, with only $O(n)$ high order terms in each neuron. Such a high order neuron of cost $O(n)$ is presented in [38] and a network utilizing such neurons is able to solve certain problems (e.g. XOR) with high success rates, while retaining the lowered cost. Various strategies to reduce the connectivity requirements and curb the proliferation of high order terms are discussed in [31, 21]. (See also [15] for an analysis of sparse Hopfield nets.)

5.2 Augmenting the cost-performance of the Hamming network

The Hamming network calculates the Hamming distance between the input pattern and each memory pattern, and selects the memory with the smallest distance, which is declared 'the winner'. This network is the most straightforward associative memory. Originally presented in [33, 32, 34], it has received renewed attention in recent years [18, 2, 19]. The Hamming network operates on binary vectors of ± 1 and is depicted in *Fig. 2*.

It is composed of two subnets. The lower subnet, denoted as the *similarity* subnet calculates the Hamming distance between the input vector and each memory pattern. It consists of two layers: An n-neuron input layer representing n-bit input patterns, and an m-neuron memory layer where each neuron represents one memory. Memory storage is achieved via the connection weights entering the neuron. The upper subnet, denoted as the *winner-take-all* (*WTA*), computes the memory which is at the minimum Hamming distance from the input. It consists of a fully connected m-neuron topology. The similarity subnet is feedforward whereas the *WTA* subnet is iterative.

The cost and performance of the Hamming network are primarily due to the *WTA* subnet. To see this, note that the number of memory patterns (i.e. memory layer neurons) thay may be stored is practically unlimited [5], and when the input patterns are distorted memory patterns, m is exponential in n. Also the operation time of the net is due to the iterative *WTA* subnet. The cost and storage measures of the net are: $CO_{wta} = O(m^2)$, $STO_{Ham} = O(m)$. We denote the operation time of the *WTA* subnet (also the total operation time of the net) by TI_{Ham}.

In aiming to increase cost-performance we wish to improve the implementation of the *WTA* network. A straightforward algorithm for finding the maximum value uses only $O(m \log m)$ basic components. We use a binary tree of analog voltage comparators [22, 27], i.e. m comparators in the first layer, $m/2$ in the second layer etc. This algorithm has been applied to Hamming nets in [5] and is of cost $O(m \log m)$ (the comparators are essentially modified operational amplifiers. Note also that there are exactly i wires (zero-valued resistors) between layer i and layer $i + 1$). Thus the cost of the reduced *WTA* subnet is $CO_r = O(m \log m)$.

The storage capacities of both networks is m. The operation time of the reduced cost *WTA* subnet is $TI_r = O(\log m)$ (the learn phase time is $O(1)$ for both nets). Our goal is to improve the cost-performance of the network, i.e. $COP_r > COP_{Ham}$.

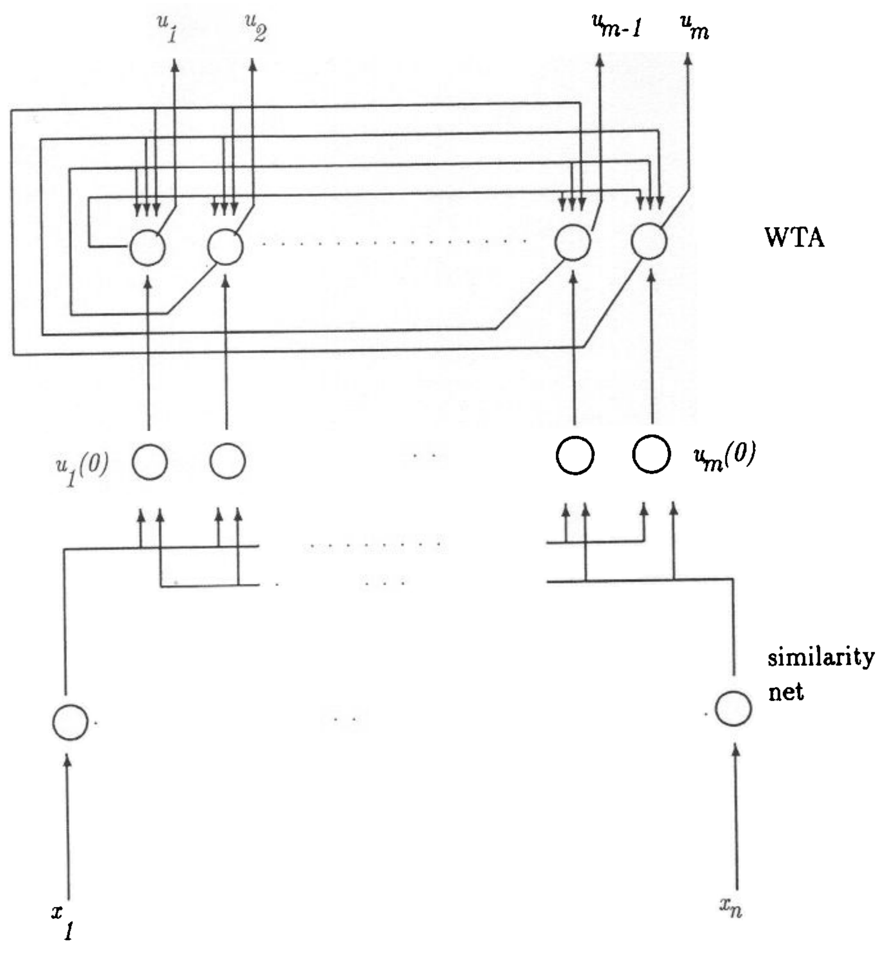

Fig. 2. A Hamming net.

$$\frac{1}{\log^2 m} > \frac{1}{mTI_{Ham}}$$

and

$$TI_{Ham} > \frac{\log^2 m}{m} .$$

For all practical purposes $\frac{\log^2 m}{m} < 1$ and thus we conclude that the proposed reduced *WTA* subnet does indeed improve the cost-performance of the Hamming network.

It is possible to improve the above result even further adapting (in terms of the necessary hardware) the algorithm presented in [36]. We obtain $TI_r = O\big(\log(\log m)\big)$ and $CO_r = O(m)$. Thus we have $COP_r = \frac{1}{\log(\log m)}$ and $COP_r > COP_{Ham}$ results in:

$$\frac{1}{\log(\log m)} > \frac{1}{mTI_{Ham}}$$

and

$$TI_{Ham} > \frac{\log(\log m)}{m} .$$

5.3 Comparing the Hopfield network and the Hamming network

Our final example involves the original Hamming network (Section 5.2) and the Hopfield network [13] whose evaluation yields:

$$CO_{Hop} = O(n^2)$$

$$STO_{Hop} = O\Big(\frac{n}{\log n}\Big)$$

$$TI_{Hop} = O\Big(\log(\log n)\Big)$$

$$COP_{Hop} = O\Big(\frac{1}{n\log(n)\log(\log n)}\Big) .$$

(See [3, 1] for the storage capacity of a Hopfield network and [15] for the time.)
The Hamming network evaluation is:

$$CO_{Ham} = O(mn + m^2)$$

$$STO_{Ham} = O(m)$$

$$COP_{Ham} = O\Big(\frac{m}{TI_{Ham}(mn + m^2)}\Big) .$$

For comparison purposes we equate $n = m$ and obtain: $COP_{Ham} = O\Big(\frac{1}{nTI_{Ham}}\Big)$. TI_{Ham} (the convergence rate of the Hamming net) has no analytical value, but rather an empirical finding obtained through simulations [18]. For networks of size $n = 1000$ the Hamming net was found to converge in approximately 10 steps. Note that for such n's, the value of $\log(n)\log(\log n)$ is 30. Thus the total cost-performance of the networks is equal.

6. Conclusions

In this paper we introduced a model for evaluating the cost-performance of analog neural networks in the extended framework of generalized high order networks. Our cost-performance model combines a *cost* measure and a *performance* measure to derive a total *cost-performance* (*COP*) measure. We defined a basic component set, and then used the total weighted component count as the cost of a net. The performance evaluation combines two basic sub-measures, namely the operation time of the net and its storage capacity.

Our model was demonstrated on some well-known networks. We compared a multi- layer first order net and a second order net, two models which are applied to problems such as symmetry, parity and contiguity. Our analysis provided an expression which states explicitly the break-even point. We then analyzed an augmented Hamming network, utilizing a more efficient *WTA* subnet, to conclude that its cost-performance was improved. Yet another example demonstrated the equivalence of the Hamming net and the Hopfield net in terms of cost-performance.

Although we concentrated in this paper on one type of analog implementation (the resistive type), others (such as storing each weight as a charge package on a capacitor [10, 11, 16]) yield themselves to the same methodology of analysis. Another issue, which is usually ignored in neural network research, is the high fan-in and fan-out of a biological neuron as opposed to electronic components [10]. The incorporation of fan-in and fan-out restrictions is straightforward in our model.

In conclusion, we have presented in this paper a model which enables the analysis of generalized high order networks by providing explicit expressions of their cost-performance. This model provides a common framework for theoretically evaluating analog networks.

Acknowledgement

We wish to thank Uri Zwick for his helpful comments in the preparation of this paper.

References

[1] D.J. Amit, H. Gutfreund and H. Sompolinsky, Storing infinite numbers of patterns in a spin-glass model of neural networks, *Physical Rev. Lett.* 55 (1985) 1530.

[2] E.E. Baum, J. Moody and F. Wilczek, Internal representations for associative memory, *Biol. Cybernet.* 59 (1987) 217–228.

[3] H.H. Chen, Y.C. Lee, G.Z. Sun, H.Y. Lee, T. Maxwell and C.L. Giles, High order correlation model for associative memory, in: J.S. Denker, ed., *AIP Conf. Proc. 151: Neural Networks for Computing* (Snowbird, Utah, 1986) 86–99.

[4] C.A. Desoer and E.S. Kuh, *Basic Circuit Theory* (McGraw-Hill, New York, 1969).

[5] E. Domany and H. Orland, A maximum overlap neural network for pattern recognition, *Physics Letters A* 125 (1) (1987) 32–34.

[6] J.A. Feldman, Dynamic connections in neural networks, *Biol. Cybernet.* 46 (1982) 27–39.

[7] D. Ferster and C. Koch, Neuronal connections underlying orientation selectivity in cat visual cortex, *trends Neurosci.* 10 (12) (Dec. 1987) 487–492.

[8] S.Y. Foo, L.R. Anderson and Y. Takefuji, Analog components for the VLSI of neural networks, *IEEE Circuits Devices Mag.* (Jul. 1990) 18–26.

[9] C.L. Giles and T. Maxwell, Learning, invariance, and generalization in high order neural networks, *Applied Optics* 26 (23) (1987) 4972–4978.

[10] H.P. Graf and L.D. Jackel, Analog electronic neural network circuits, *IEEE Circuits Devices Mag.* (Jul. 1989) 44–49.

[11] H.P. Graf, E. Sackinger, B. Boser and L.D. Jackel, Recent developments of electronic neural nets in the US and Canada, in: *Proc. 2nd Internat. Conf. on Microelectronics for Neural Networks*, Munich, Germany (Oct. 1991) 471–488.

[12] R. Hecht-Nielsen, *Neurocomputing* (Addison-Wesley, reading, MA, 1990).

[13] J.J. Hopfield, Neural networks and physical systems with emergent collective computational abilities, *Proc. Nat. Acad. Sci. USA* 79 (Apr. 1982) 2554–2558.

[14] C. Koch, A network model for cortical orientation selectivity in cat striate cortex, *Invest. Ophthalmol. Vis. Sci.* 28 (3) (Suppl) (1987) 126.

[15] J. Komlós and R. Paturi, Effect of connectivity in associative memory models, in: *Proc. 29th IEEE Annual Symp. on Foundations of Computer Science* (1988) 138–147.

[16] R. Lauwereins and J. Bruck, Efficient implementation of a neural multiplier, in: *Proc. 2nd Internat. Conf. on Microelectronics for Neural Networks*, Munich, Germany (Oct. 1991) 217–230.

[17] F.T. Leighton, *Complexity Issues in VLSI* (MIT Press, Cambridge, MA, 1983).

[18] R.P. Lippmann, B. Gold and M.L. Malpass, A comparison of Hamming and Hopfield neural nets for pattern classification, Technical Report TR-769, MIT Lincoln Laboratory, 1987.

[19] R.P. Lippmann, An introduction to computing with neural nets, *IEEE ASSP Mag.* (1987) 4–22.

[20] R.P. Lippmann, Pattern classification using neural networks, *IEEE Commun. Mag.* (Nov. 1989) 47–64.

[21] T. Maxwell, C.L. Giles, Y.C. Lee and H.H. Chen, Nonlinear dynamics of artificial neural systems, in: J.S. Denker, ed., *AIP Conference Proceedings 151: Neural Networks for Computing* (Snowbird, Utah, 1986) 299–305.

[22] J. Millman and A. Grabel, *Microelectronis*, 2nd ed. (McGraw-Hill, New York, 1987).

[23] P. Mueller, T. Martin and F. Putzrath, General principles of operations in neuron nets with application to acoustical pattern recognition, in: E.E. Bernard and M.R. Kare, eds., *Biological Prototypes and Synthetic Systems*, vol. 1 (Plenum, New York, 1962) 192–212.

[24] A.F. Murray, D. Del Corso and L. Tarassenko, Pulse-stream VLSI neural networks mixing analog and digital techniques, *IEEE Trans. Neural Networks* 2 (2) (Mar. 1991) 193–204.

[25] P. Peretto and J.J. Niez, Long term memory storage capacity of multiconnected neural networks, *Biol. Cybernet.* 54 (1986) 53–63.

[26] D.E. Rumelhart, G.E. Hinton and R.J. Williams, Learning internal representations by error propagation, in: D.E. Rumelhart, J.L. McClelland and the PDP Research Group, eds., *Parallel Distributed Processing, Vol. 1: Foundations* (MIT Press, Cambridge, MA, 1986) 318–362.

[27] C.J. Savant, M.S. Roden and G.L. Carpenter, *Electronic Circuit Design – An Engineering Approach* (Benjamin/Cummings, Menlo Park, CA, 1987).

[28] T.J. Sejnowski, C. Koch and P.S. Churchland, Computatinal neuroscience, *Science* 241 (Sep. 1988) 1299–1306.

[29] T.J. Sejnowski, High-order Boltzman machines, in: J.S. Denker, ed., *AIP Conf. Proc. 151: Neural Networks for Computing* (Snowbird, Utah, 1986) 398–402.

[30] H.T. Siegelmann and E.D. Sontag, On the computational power of neural nets, Technical Report SYCON-91-11, Rutgers Center for Systems and Control, Rutgers University, New Brunswick, NJ, November 1991.

[31] L. Spirkovska and M.B. Reid, Connectivity strategies for higher-order neural networks applied to pattern recognition, in: *Proc. Internat. Joint Conf. on Neural Networks*, vol. I, San Diego, CA (Jun. 1990) 21–26.

[32] K. Steinbuch and U.A.W. Piske, Learning matrices and their applications, *IEEE Trans. Electronic Comput.* (1963) 846–862.

[33] K. Steinbuch, Die Lernmatrix, *Kybernetik* 1 (1961) 36–45.

[34] W.K. Taylor, Cortico-thalamic organization and memory, *Proc. Royal Soc. London B* 159 (1964) 466–478.

[35] J.D. Ullman, *Computational Aspects of VLSI* (Computer Science Press, Rockville, MD, 1984).

[36] L.G. Valiant, Parallelism in comparison models, *SIAM J. Comp.* 4 (1975) 348–355.

[37] B. Widrow and M.E. Hoff, Adaptive switching circuits, in: *IRE WESCON Convention Record* (New York, 1960) 96–104, also in: J.A. Anderson and E. Rosenfeld, eds., *Neurocomputing* (MIT Press, Cambridge, 1988) 126–134.

[38] H. Yang and C.C. Guest, High order neural networks with reduced numbers of interconnection weights, in: *Proc. Internat. Joint Conf. on Neural Networks*, vol. III, San Diego, CA (Jun. 1990) 281–286.

Yehezl
a resea
Scienc

Moshe Sipper received the B.A. degree from the Technion – Israel Institute of Technology in 1985 and the M.Sc. degree from Tel Aviv University in 1989, both in Computer Science. During the years 1985–91 he was a Software Engineer in the Israel Defense Forces. Currently he is a Ph.D. student at Tel Aviv University. His research interests are the analysis and evaluation of neural network models.

Yehezkel Yeshurun received his Ph.D. at Tel Aviv University in Applied Mathematics. During 1986–1987 he was a research scientist at New York University Brain Research Labs. He is now with the Department of Computer Science at Tel Aviv University. His research interests are in Computational Vision and Neural Modeling.