LA-UR- 98-3325

Approved for public release; distribution is unlimited.

Title: OPTIMAL NEURAL COMPUTATIONS REQUIRE ANALOG PROCESSORS

CONF-9809118--

Author(s): Valeriu C. Beiu, NIS-1

Submitted to: Invited plenary talk at the International Conference on parallel computing in Electrical Engineering Bialystok, Poland, September 2-5, 1998

DISTRIBUTION OF THIS DOCUMENT IS UNLIMITED

MASTER

Los Alamos
NATIONAL LABORATORY

DISCLAIMER

Optimal Neural Computations Require Analog Processors

Valeriu Beiu *
Division Space and Atmospheric Sciences NIS-1, Los Alamos National Laboratory
Los Alamos, MS D466, NM 87545, USA
E-mail: beiu@lanl.gov

Abstract — **This paper discusses some of the limitations of hardware implementations of neural networks. We start by presenting neural structures and their biological inspirations, while mentioning the simplifications leading to artificial neural networks. Further the focus will be on hardware imposed constraints. We will present recent results for three different alternatives of parallel implementations of neural networks: digital circuits, threshold gate circuits, and analog circuits. The *area* and the *delay* will be related to the neurons' *fan-in* and to the precision of their synaptic *weights*. The main conclusion is that hardware-efficient solutions require analog computations, and suggests the following two alternatives: (*i*) cope with the limitations imposed by silicon, by speeding up the computation of the elementary 'silicon' neurons; (*ii*) investigate solutions which would allow the use of the third dimension (*e.g.* using optical interconnections).**

Keywords — **neural networks, Boolean functions, threshold gates, analog circuits, circuit complexity, VLSI complexity, *fan-in*, *size*, precision (accuracy).**

I. Introduction

The model we shall discuss wants to duplicate the activity of the human brain. This is made of living neurons composed of a cell body and many outgrowths. One of these is the *axon* — which may branch into several collaterals. The axon is the 'output' of the neuron. The other outgrowths are the *dendrites*. The end of the axons from other neurons are connecting to the dendrites through 'spines'. Active pumps in the nerve cell walls push sodium ions outside, while keeping fewer potassium ions inside. Therefore, their tendency is to keep the cell body at a small negative electric potential (–60mV). The electrical balance varies at the exit point of the axon. If the electrical potential of the cell becomes too positive (+10÷15mV), the potential suddenly jumps to about +60mV. After a short delay of 2÷3ms the potential returns to the normal negative value (–60mV). This change of potentials is sequential and is called an *action potential*. The action potential travels down the axon and its branches (with a speed in the range 1÷10m/s). This variation of potential represents the signal sent by one neuron to its neighbours. The generation of the signal is achieved by summing the signals coming from the dendrites. The strength of the action potentials travelling along an axon are identical, nevertheless, the effects to the neighbouring cells are different. This is due to the rescaling effect which takes place at the *synapse*. Although over-simplified, this description of the living nerve cells is a correct representation of the system.

Formally, a *network* is an acyclic graph having several input nodes, and some (at least one) output nodes. If a synaptic *weight* is associated with each edge, and each node computes the *weighted sum* of its inputs to which a nonlinear activation function is then applied:

$$f(x) = f(x_1, \ldots, x_\Delta) = \sigma\left(\sum_{i=1}^{\Delta} w_i x_i + \theta\right), \qquad (1)$$

the network is a *neural network* (NN), with $w_i \in \mathbb{R}$ the synaptic *weights*, $\theta \in \mathbb{R}$ known as the *threshold*, Δ being the *fan-in*, and σ a non-linear activation function. Because the underlying graph is acyclic, the network does not have feedback, and can be layered. That is why such a network is also known as a *multilayer feedforward neural network*. The connection *weights* are quite important, as it is their modification that allows the NN to 'learn'. The basic idea is to present the examples to the NN and change the *weights* in such a way as to improve the results (*i.e.*, the outputs of the NN will be 'closer' to the desired values). The cost functions used to characterise a NN are:

- *depth* (*i.e.*, number of edges on the longest input-to-output path, or number of layers); and
- *size* (*i.e.*, number of neurons).

In the last decade the tremendous impetus of VLSI technology has made neurocomputer design a really lively research topic. Hundreds of designs have been already build, while several are available as commercial products. Still, we are far from the main objective as can be clearly seen from Fig. 1 where the horizontal axis represents the number of synapses (*i.e.*, the connectivity), while the vertical axis represents the 'power of computation' in *connections per second* (CPS). It becomes clear that biological NNs are far ahead of digital, analog and even future optical implementations. This paper will try to explain why this is the case.

* On leave of absence from the "Politehnica" University of Bucharest, Computer Science Department, Spl. Independenței 313, RO-77206 Bucharest, România.

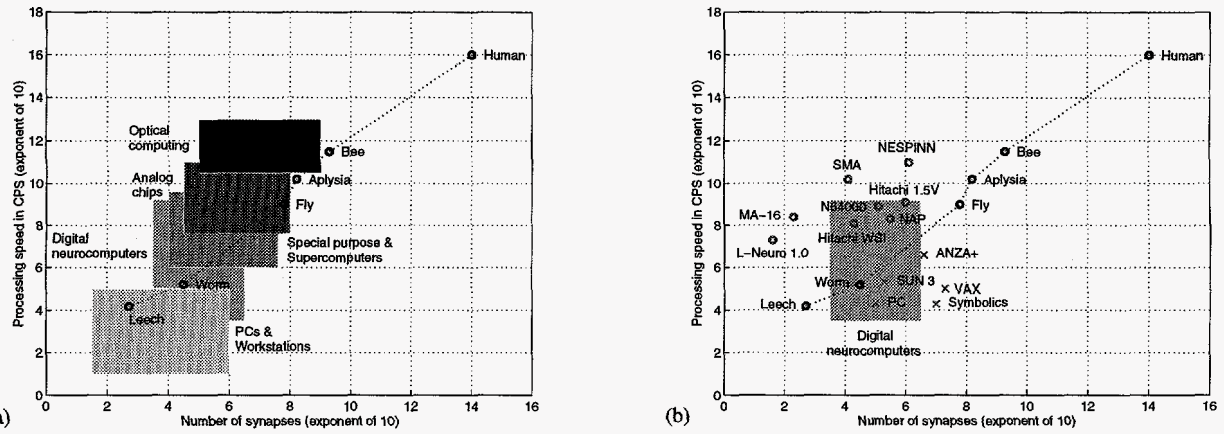

Fig. 1. Different hardware alternatives for implementing artificial neural networks: (a) an enhanced version from [39]; (b) digital neurochips as circles, and classical computers as crosses (for more details see [10]).

For hardware implementations the *area* of the connections counts, and the *area* of one neuron can be related to its associated *weights*, thus *"comparing the number of nodes is inadequate for comparing the complexity of NNs as the nodes themselves could implement quite complex functions"* [99]. That is why several authors have taken into account other cost functions, which can be linked to VLSI by the assumptions one makes on how the *area* of a chip scales with the *weights* and the *thresholds* [9, 10, 15, 16]. Here are some of the other measures (*i.e.*, 'cost functions') — beside *size* — which have already been used:

- the total *number-of-connections*, or Σ_{NN} *fan-ins*, has been used by several authors [1, 40, 67, 80];
- the total *number-of-bits* needed to represent the *weights* and *thresholds* $\Sigma_{NN}(\Sigma_i \lceil \log |w_i| \rceil + \lceil \log |\theta| \rceil)$ [1] has been used by others [29, 99];
- the sum of all the absolute values of the *weights* and *thresholds* $\Sigma_{NN}(\Sigma_i |w_i| + |\theta|)$ has also been advocated [9, 16, 23, 25, 26, 27], while another similar 'cost function' is $\Sigma_{NN}(\Sigma_i w_i^2 + \theta^2)$, which has been used in the context of genetic programming for reaching minimal NNs [101].

The sum of all the absolute values of the *weights* and *thresholds* has been used as an optimum criterion:

- for linear programming synthesis [71];
- for defining the minimum-integer *threshold gate* (TG) realisation of a *Boolean function* (BF) [50];
- in the context of computational learning theory for improving on several VC-theory bounds [6]; in particular, the measure was used under the name of *"total weight magnitude,"* and it was proven that the generalisation error of NNs used for classification depends on the size of the *weights* — rather than the number of *weights* — by showing that the misclassification probability converges at a rate of $O\{(cA)^l/\sqrt{m}\}$ (here A is the sum of the magnitude of the *weights*, l is the *depth*, m is the number of examples, and c is a constant).

[1] In this paper $\lceil x \rceil$ is the ceiling of x, *i.e.*, the smallest integer greater than or equal to x, and $\lfloor x \rfloor$ is the floor of x, *i.e.*, the largest integer less than or equal to x, and all the logarithms are taken to base 2 (except mentioned otherwise).

Such approximations can easily be related to assumptions on how the *area* of a chip scales with the *weights* and the *thresholds* [10, 12, 15, 21]:

- for digital implementation, the *area* scales with the cumulative storage of *weights* and *thresholds* (as the bits for representing those *weights* and *thresholds* have to be stored);
- for analog implementations (*e.g.*, using resistors or capacitors) the same type of scaling is valid (although it is possible to come up with implementations having binary encoding — for which the *area* would scale with the cumulative log-scale size of the parameters);
- some types of implementations (*e.g.*, transconductance ones) even offer a constant size per element, thus in principle scaling only with the number of parameters (*i.e.*, with the total *number-of-connections*).

With respect to *delay*, two VLSI models have been commonly in use [96]:

- the simplest one assumes that *delay* is proportional to the input capacitance, hence a TG introduces a *delay* proportional to its *fan-in*;
- a more exact one considers the capacitance along any wire, hence the *delay* is proportional to the *length* of the connecting wires.

It is worth emphasising that it is anyhow desirable to limit the range of parameter values for VLSI implementations [100], be they digital or analog, because: (*i*) the maximum value of the *fan-in* [40, 98]; and (*ii*) the maximal ratio between the largest and the smallest *weight* cannot grow over a certain (technological) limit [29, 34].

The paper starts by overviewing several results dealing with the approximation capabilities of NNs, and details upper and lower bounds on the *size* of *threshold gate circuits* (TGCs). These are followed by solutions which are optimal with respect to different cost functions. We show that both Boolean and TGCs require exponential *size* for implementing arbitrary BFs, while there are TGCs which have low precision and small *fan-ins*. Further, we argue that *size*-optimal solutions of discrete NNs can be obtained only in analog circuitry, but require very high precision and large *fan-ins* (based on a fresh

constructive solution for Kolmogorov's superpositions). It follows that the mapping onto silicon — lacking the third dimension of the biological nets — translates into limited *fan-in* and reduced precision. Several conclusions are ending the paper.

II. Previous Results

NNs have been experimentally shown to be quite effective in many applications (see *Applications of Neural Networks* in [3], together with *Part F: Applications of Neural Computation* and *Part G: Neural Networks in Practice: Case Studies* from [35]). This success has led researchers to undertake a rigorous analysis of their mathematical properties and has generated two directions of research for finding:

- existence/constructive proofs for the *'universal approximation problem'*;
- tight bounds on the *size* of the NNs solving the approximation problem.

Both aspects will be shortly discussed further.

A. Neural Networks as Universal Approximators

One line of research has concentrated on the approximation capabilities of NNs [28, 53]. It was started in 1987 by Hecht-Nielsen [43] and Lippmann [65] who, together with LeCun [63], were probably the first to recognise that the specific format from [89, 90] of the form:

$$f(x_1, \ldots, x_n) = \sum_{q=1}^{2n+1} \{\Phi_q [\sum_{p=1}^{n} \alpha_p \psi (x_p + qa)]\} \quad (2)$$

of Kolmogorov's superpositions $f(x_1, \ldots, x_n) = \sum_{q=1}^{2n+1} \Phi_q (y_q)$ [59], can be interpreted as a NN with one hidden layer. This gave an existence proof of the approximation properties of NNs. The first nonconstructive proof was given in 1988 by Cybenko [31, 32] using a continuous activation function, and was independently presented by Irie and Miyake [52]. Similar results for radial basis functions were shortly reported [41, 81]. Thus, the fact that NNs are computationally universal — with more or less restrictive conditions — when modifiable connections are allowed, was established. Different enhancements have been later presented (for more details see [16, 85]):

- Funahashi [36] proved the same result in a more constructive way, and refined the use of Kolmogorov's theorem in [43], giving an approximation result for two-hidden-layer NNs;
- Hornik *et al.* [48] showed that the continuity requirement for the output function can partly be removed;
- Hornik *et al.* [49] also proved that a NN can approximate simultaneously a function and its derivative;
- Park and Sandberg [77, 78] used radial basis functions in the hidden layer, and gave an almost constructive proof;
- Hornik [46] showed that the continuity requirement can be completely removed, the activation function having to be 'bounded and nonconstant';
- Geva and Sitte [38] proved that four-layered NNs with sigmoid activation function are universal approximators;
- Kůrková [61] and Kůrková *et al.* [62] have demonstrated the existence of approximate superposition representations within the constraints of NNs, *i.e.* ψ and Φ_q can be approximated by $\sum a_r \sigma (b_r x + c_r)$, where σ is an arbitrary activation sigmoidal function (depending on approximation, the *size* of the resulting NNs is between $nm(m+1)$ and $m^2(m+1)^n$);
- Mhaskar and Micchelli [68, 69] approach was based on the Fourier series of the function, by truncating the infinite sum to a finite set, and rewriting e^{ikx} in terms of the activation function (which now has to be periodic);
- Koiran [58] presented a new proof on the line of Funahashi's [36], but more general in that it allows the use of units with 'piecewise continuous' activation functions;
- Leshno *et al.* [64] relaxed the condition for the activation function to 'locally bounded piecewise continuous' (*i.e.*, if and only if the activation function is not a polynomial), thus embedding as special cases almost all the activation functions that have been previously reported in the literature;
- Hornik [47] later proved that: (*i*) if the activation function is locally Riemann integrable and nonpolynomial, the *weights* and the *thresholds* can be constrained to arbitrarily small sets; and (*ii*) if the activation function is locally analytic, a single universal *threshold* will do;
- Funahashi and Nakamura [37] showed that the universal approximation theorem also holds for trajectories;
- Sprecher [91] has demonstrated that there are universal hidden layers that are independent of n;
- Barron [5] described spaces of functions that can be approximated by the relaxed algorithm of Jones [55] using functions computed by single-hidden-layer NNs;
- Ito [54] gave an elementary constructive method improving on the estimates of Kůrková [61], the *size* of the resulting NNs being now between nm and m^n.

These results — with the exception of [5, 58, 77, 78] — were obtained *"provided that sufficiently many hidden units are available"* (*i.e.*, with no claims on the *size* minimality). More constructive solutions have been obtained in very small *depth* later [61, 54, 56, 74, 75], but their *size* — *or the required precision* — grows fast with respect to the number of dimensions n.

Two important recent results are those of:

- Attali and Pagès [4], who have given an elementary proof based on the Taylor expansion and the Vandermonde determinant, yielding bounds for the design of the hidden layer and convergence results for the derivatives;
- Sprecher [92–94], who gave an explicit numerical algorithm for superpositions.

B. Threshold Gate Circuits

The other line of research was to find the smallest *size* NN which can realise an arbitrary function given a set of m vectors from $\mathbb{R}^n$. Many results have been obtained for TGs [71]. The first lower bound on the *size* of a TGC for "almost all" n-ary BFs ($f: \mathbb{B}^n \rightarrow \mathbb{B}$) was given by Neciporuk [73]:

$$size \geq 2(2^n/n)^{1/2}. \quad (3)$$

Later a very tight upper bound was proven in *depth* = 4 [66]:

$$size \leq 2\,(2^n/n)^{1/2} \times \{1 + \Omega\,[(2^n/n)^{1/2}]\}. \qquad (4)$$

A similar existence exponential lower bound of $\Omega\,(2^{n/3})$ for arbitrary BFs can be found in [87], which also gives bounds for many particular but important BFs (see also [84]).

For classification problems ($f: \mathbb{R}^n \rightarrow \mathbb{B}^k$), the first result was that a NN of *depth* = 3 and *size* = $m-1$ could compute an arbitrary dichotomy. The main improvements have been:

- Baum [7] presented a TGC with one hidden layer having $\lceil m/n \rceil$ neurons capable of realising an arbitrary dichotomy on a set of m points *in general position* in $\mathbb{R}^n$; if the points are on the corners of the n-dimensional hypercube, $m-1$ nodes are still needed;
- a slightly tighter bound of only $\lceil 1 + (m-2)/n \rceil$ neurons in the hidden layer for realising an arbitrary dichotomy on a set of m points (which satisfy a more relaxed topological assumption) was proven in [51]; the $m-1$ nodes condition was shown to be the least upper bound needed;
- Arai [2] showed that $m-1$ hidden neurons are necessary for arbitrary separability, but improved the bound for the dichotomy problem to $m/3$ (without any condition);
- Beiu [8] has detailed the following existence lower and upper bounds: $2m\log m/n^2 < size < 2m\log m/n^2 \log n$, by estimating the entropy of the data-set;
- Beiu and De Pauw [17] have presented several improvements on the previous results [8], by proving two new bounds $2m/(n\log n) < size < 1.44m/n$ (see also [18, 24]).

Other existence lower bounds for the arbitrary dichotomy problem [42, 79] are:

- a *depth*-2 TGC requires $m/\{n \log(m/n)\}$ TGs;
- a *depth*-3 TGC requires $2\,(m/\log m)^{1/2}$ TGs in each of the two hidden layer (if $m \gg n^2$);
- an arbitrarily interconnected TGC without feedback needs $(2m/\log m)^{1/2}$ TGs (if $m \gg n^2$).

One study [30] has tried to unify these two lines of research by first presenting analytical solutions for the general NN problem in one dimension (having infinite *size*), and then giving practical solutions for the one-dimensional cases (*i.e.*, including an upper bound on the *size*). Extensions to the n-dimensional case using three- and four-layers solutions were derived under piecewise constant approximations, and under piecewise linear approximations (using ramps instead of sigmoids).

C. Boolean Functions

The particular case of BFs has been intensively studied [16, 76]. Many results have been obtained for particular BFs [84, 87]. For example, $\mathbb{F}_{n,m}$ is the class of BFs of n variables having m groups of ones in their truth table. Obviously, any BF can be represented by a suitable collection of its true values (ones), but for achieving that the number of groups of ones grows exponentially (*i.e.*, $\mathbb{F}_{n,2^{n/2}}$ completely covers $\mathbb{B}_n$, the set of all n-ary BFs). This class of functions has been introduced and analysed by Red'kin [83], who constructively proved a *size*-optimal result in *depth* = 3.

Proposition 1 (from [83]) The complexity realisation (i.e., number of threshold elements) of $\mathbb{F}_{n,m}$ (the class of Boolean functions $f(x_1, x_2, \ldots, x_{n-1}, x_n)$ that have exactly m groups of ones) is at most $2\,(2m)^{1/2} + 3$.

The construction has: a first layer of $\lceil (2m)^{1/2} \rceil$ TGs (COMPARISONs) with *fan-in* = n and *weights* $\leq 2^{n-1}$; a second layer of $2\lceil (m/2)^{1/2} \rceil$ TGs of *fan-in* = $n + \lceil (2m)^{1/2} \rceil$ and *weights* $\leq 2^n$; a TG of *fan-in* = $2\lceil (m/2)^{1/2} \rceil$ and *weights* $\in \{-1, +1\}$ in the third layer. This result is valid for unlimited *fan-in* TGs. Red'kin also proved that if the implementation of BFs of this type is restricted to circuits having no more than three layers, than the upper bound — following his method of synthesis — is equal to the lower bound obtained from capacity considerations. Although this construction is *size*-optimal, it is not VLSI-optimal as the *weights* and *thresholds* grow exponentially with the number of inputs (while the *fan-in* is polynomial).

A general solution for synthesising one BF with *fan-in* 2 AND-OR gates is based on the classical construction developed by Shannon [86]. It was later extended to the multioutput case, and modified to apply to NN by Horne and Hush [45]:

Proposition 2 (from [45]) Arbitrary Boolean functions of the form $f: \{0,1\}^n \rightarrow \{0,1\}^\mu$ can be implemented in a neural network of perceptrons restricted to fan-in 2 with a node complexity of $\Theta\,\{\mu\, 2^n/(n + \log\mu)\}$ and requiring $\boldsymbol{O}\,(n)$ layers.

Proof Decompose each output BF into two subfunctions using Shannon's decomposition [86]:

$$f(x_1, x_2, \ldots, x_{n-1}, x_n) = \bar{x}_1 f_0\,(x_2, \ldots, x_{n-1}, x_n) + x_1 f_1\,(x_2, \ldots, x_{n-1}, x_n).$$

By doing this recursively, the output BFs will be implemented by binary trees. To eliminate most of the lower level nodes, replace them with a subnetwork that computes *all the possible BFs* needed by the higher level nodes. Each subcircuit eliminates one variable and has three nodes (one OR and two ANDs). Thus, the upper tree has:

$$size_{upper} = 3\,\mu \cdot \sum_{i=0}^{n-q-1} 2^i = 3\,\mu\,(2^{n-q} - 1), \qquad (5)$$

and $depth_{upper} = 2\,(n-q)$. The subfunctions now depend on q variables, and the lower subnetwork that computes all the possible BFs of q variables has:

$$size_{lower} = 3 \cdot \sum_{i=1}^{q} 2^{2^i} < 4 \cdot 2^{2^q}, \qquad (6)$$

and $depth_{lower} = 2\,q$ (see Fig. 2 from [45]).

That q which minimises the *size* of the two subnetworks:

$$size_{BFs} = size_{upper} + size_{lower} \qquad (7)$$

is determined by solving $d(size_{BFs})/dq = 0$, and gives:

$$q \approx \log\{n + \log\mu - 2\log(n + \log\mu)\}. \quad (8)$$

By substituting (8) in (5) and (6), the minimum *size*:

$$size^*_{BFs}(n, \mu) \approx 3\mu \cdot 2^{n-q} = 3\mu \cdot 2^n/(n + \log\mu) \quad (9)$$

is determined. ❑

III. Hardware optimal solutions

It is well known that implementing arbitrary BFs using classical Boolean gates (*i.e.*, AND and OR gates) requires exponential *size* circuits. As has been presented in the previous section, the known bounds for *size* are also exponential if TGCs are used to solve arbitrary BFs. These bounds reveal exponential gaps, and also suggest that TGCs with more layers might have a smaller *size* (*depth* ≠ small const. [11, 12, 19]).

A. Boolean Functions Using Threshold Gates

We start from the classical construction developed by Shannon [86] for synthesising one BF with *fan-in* 2 AND-OR gates, and generalise *Proposition 2* (from [45]) to arbitrary *fan-in*.

Proposition 3 (from [19]) Arbitrary Boolean functions $f: \{0, 1\}^n \to \{0, 1\}^\mu$ *can be implemented in a neural network of perceptrons restricted to fan-in* Δ *in* $\boldsymbol{O}(n/\log\Delta)$ *layers.*

Proof We use the approach of Horne & Hush [45] and limit the *fan-in* to Δ. Each output BF can be decomposed in $2^{\Delta-1}$ subfunctions (*i.e.*, $2^{\Delta-1}$ AND gates). The OR gate would have $2^{\Delta-1}$ inputs. Thus, we have to decompose it in a Δ-ary tree of *fan-in* $= \Delta$ OR gates. This first decomposition step eliminates $\Delta - 1$ variables and generates a tree of:

$$depth = 1 + \lceil(\Delta - 1)/\log\Delta\rceil,$$
$$size = 2^{\Delta-1} + \lceil(2^{\Delta-1} - 1)/(\Delta - 1)\rceil.$$

Repeating this procedure recursively k times, we have:

$$depth_{upper} = k \cdot \{1 + \lceil(\Delta - 1)/\log\Delta\rceil\} \quad (10)$$

$$size_{upper} = \{2^{\Delta-1} + \lceil(2^{\Delta-1} - 1)/(\Delta - 1)\rceil\} \cdot \sum_{i=0}^{k-1} 2^{i(\Delta-1)}$$
$$= size \cdot \{2^{k(\Delta-1)} - 1\}/(2^{\Delta-1} - 1) \quad (11)$$
$$\cong 2^{k(\Delta-1)}(1 + 1/\Delta)$$
$$\approx 2^{k\Delta-k},$$

where the subfunctions depend only on $q = n - k\Delta$ variables. We now generate all the possible subfunctions of q variables with a subnetwork of:

$$depth_{lower} = \lfloor(n - k\Delta)/\Delta\rfloor \cdot \{1 + \lceil(\Delta - 1)/\log\Delta\rceil\} \quad (12)$$

$$size_{lower} = \{2^{\Delta-1} + \lceil(2^{\Delta-1} - 1)/(\Delta - 1)\rceil\} \cdot \sum_{i=1}^{\lfloor n/\Delta\rfloor - k} 2^{2^{n-k\Delta-i\Delta}}$$
$$= size \cdot \{2^{2^0} + 2^{2^\Delta} + \ldots + 2^{2^{n-(k+1)\Delta}}\}$$
$$< (size + 1) \cdot 2^{2^{n-(k+1)\Delta}} \quad (13)$$
$$\approx 2^\Delta \cdot 2^{2^{n-k\Delta-\Delta}}. \quad (14)$$

The inequality (13) can be proved by induction. Clearly:

$$size \cdot 2^{2^0} < (size + 1) \cdot 2^{2^0}.$$

Let us consider the statement true for α; we shall prove it for $\alpha + 1$:

$$size \cdot \{2^{2^0} + \ldots + 2^{2^{\alpha\Delta}}\} + size \cdot 2^{2^{(\alpha+1)\Delta}} < size \cdot 2^{2^{(\alpha+1)\Delta}} + 2^{2^{(\alpha+1)\Delta}}$$
$$size \cdot \{2^{2^0} + \ldots + 2^{2^{\alpha\Delta}}\} < (size + 1) \cdot 2^{2^{\alpha\Delta}}$$

(due to hypothesis), thus:

$$(size + 1) \cdot 2^{2^{\alpha\Delta}} < 2^{2^{(\alpha+1)\Delta}}$$

and computing the logarithm of the left side:

$$2^{\alpha\Delta} + \log(size + 1) = 2^{\alpha\Delta} + \log\{2^{\Delta-1} + \lceil(2^{\Delta-1} - 1)/(\Delta - 1)\rceil\}$$
$$< 2^{\alpha\Delta} + \log\{2^{\Delta-1} + 2^{\Delta-1}/\Delta + 1\}$$
$$< 2^{\alpha\Delta} + \Delta$$
$$< 2^{(\alpha+1)\Delta}.$$

From (10) and (12) we can estimate $depth_{BFs}$:

$$depth_{BFs} = \{k + \lfloor(n - k\Delta)/\Delta\rfloor\} \cdot \{1 + \lceil(\Delta - 1)/\log\Delta\rceil\}$$
$$= (n/\Delta) \cdot (\Delta/\log\Delta + 1) \quad (15)$$
$$\approx n/\log\Delta$$
$$= \boldsymbol{O}(n/\log\Delta)$$

and from (11) and (13) $size_{BFs}$ as:

$$size_{BFs} = \mu \cdot size \cdot \{2^{k(\Delta-1)} - 1\}/(\Delta - 1) + (size + 1) \cdot 2^{2^{n-(k+1)\Delta}}$$
$$\approx \mu \cdot 2^{k\Delta-k} + 2^\Delta \cdot 2^{2^{n-k\Delta-\Delta}} \quad (16)$$

concluding the proof. ❑

Proposition 4 (from [19]) For arbitrary Boolean functions $f: \{0, 1\}^n \to \{0, 1\}^\mu$ *implemented by a neural network of perceptrons, all the critical points of the size:* $size_{BFs}(\mu, n, k, \Delta)$, *are relative minimum and are situated in the (close) vicinity of the parabola* $k\Delta \approx n - \log(n + \log\mu)$.

Proof To determine the critical points, we equate the partial derivatives to zero. Starting from the approximation of $size_{BFs}$

given by (16) we compute $\partial size_{BFs} / \partial k = 0$:

$$\mu \cdot 2^{k\Delta - k} (\ln 2)(\Delta - 1) + 2^{\Delta} 2^{2^{n-k\Delta-\Delta}} (\ln 2)\, 2^{n-k\Delta-\Delta} (\ln 2)(-\Delta) = 0$$

$$\{\mu(\Delta - 1)/\Delta/(\ln 2)\} \cdot 2^{2k\Delta - k - n} = 2^{2^{n-k\Delta-\Delta}}$$

and using the notations $k\Delta = \gamma$, $\beta = \mu(\Delta - 1)/(\Delta \ln 2)$, and taking logarithms of both sides:

$$\log\beta + 2\gamma - k - n = 2^{n-\gamma-\Delta} \tag{17}$$

which has an approximate solution $\gamma \approx n - \log(n + \log\mu)$.

The same result can be obtained by computing with finite differences (instead of approximating the partial derivative):

$$size_{BFs}(\mu, n, k+1, \Delta) - size_{BFs}(\mu, n, k, \Delta) = 0$$

$$size \cdot \left\{\mu \cdot 2^{k\Delta - k} - 2^{2^{n-k\Delta-\Delta}}\right\} = 0$$

$$\mu \cdot 2^{k\Delta - k} = 2^{2^{n-k\Delta-\Delta}}$$

and after taking twice the logarithm of both sides, and using the same notations, we have:

$$\log\{\log\mu + \gamma(1 - 1/\Delta)\} = n - \gamma - \Delta$$

$$\gamma = n - \{\Delta + \log(1 - 1/\Delta)\} - \log\{\gamma + \Delta/(\Delta - 1) \cdot \log\mu\}$$

$$\approx n - \Delta - \log(\gamma + \log\mu), \tag{18}$$

which has the same approximate solution:

$$\gamma = n - \log(n + \log\mu).$$

Starting again from (16), we compute $\partial size_{BFs} / \partial\Delta = 0$:

$$\mu\, 2^{k\Delta - k} (\ln 2)\, k + 2^{\Delta} (\ln 2)\, 2^{2^{n-k\Delta-\Delta}} + 2^{\Delta} 2^{2^{n-k\Delta-\Delta}} (\ln 2)\, 2^{n-k\Delta-\Delta} (\ln 2)(-k) = 0$$

$$\mu k \cdot 2^{\gamma - k} = k(\ln 2) \cdot 2^{n-\gamma} \cdot 2^{2^{n-\gamma-\Delta}} - 2^{\Delta} \cdot 2^{2^{n-\gamma-\Delta}}$$

$$\mu k \cdot 2^{\gamma - k} \cdot 2^{\gamma - n} = k(\ln 2) \cdot 2^{2^{n-\gamma-\Delta}} - 2^{\Delta} \cdot 2^{\gamma - n} \cdot 2^{2^{n-\gamma-\Delta}}$$

$$\mu k \cdot 2^{2\gamma - k - n} = \{k(\ln 2) - 2^{\gamma + \Delta - n}\} \cdot 2^{2^{n-\gamma-\Delta}}$$

$$(\mu/\ln 2) \cdot 2^{2\gamma - k - n} = \{1 - 2^{\gamma + \Delta - n}/(k \ln 2)\} \cdot 2^{2^{n-\gamma-\Delta}}$$

which — by neglecting $2^{\gamma+\Delta}/\{k(\ln 2) \cdot 2^{n}\}$ — gives:

$$\log\beta + 2\gamma - k - n = 2^{n-\gamma-\Delta}$$

i.e., the same equation as (17).

These show that the critical points are situated in the (close) vicinity of the parabola $k\Delta \approx n - \log(n + \log\mu)$. ❑

From *Proposition 4* it follows that *size*-optimal TGCs can be obtained for small *fan-ins* (*i.e.*, from constant to at most $n - \log n < n$). The exact *size*:

$$size_{BFs} = size_{lower} + \mu \cdot size_{upper}$$

has been computed for many different values of n, μ, Δ and k. Some results of those simulations are plotted in Fig. 2. From Fig. 2(a), 2(b) and 2(c) it seems that k and Δ have roughly the same influence (on $size_{BFs}$). The discrete parabola-like curves which are approximations of $k\Delta \approx n - \log(n + \log\mu)$ can be seen in Fig. 2(d), 2(e) and 2(f).

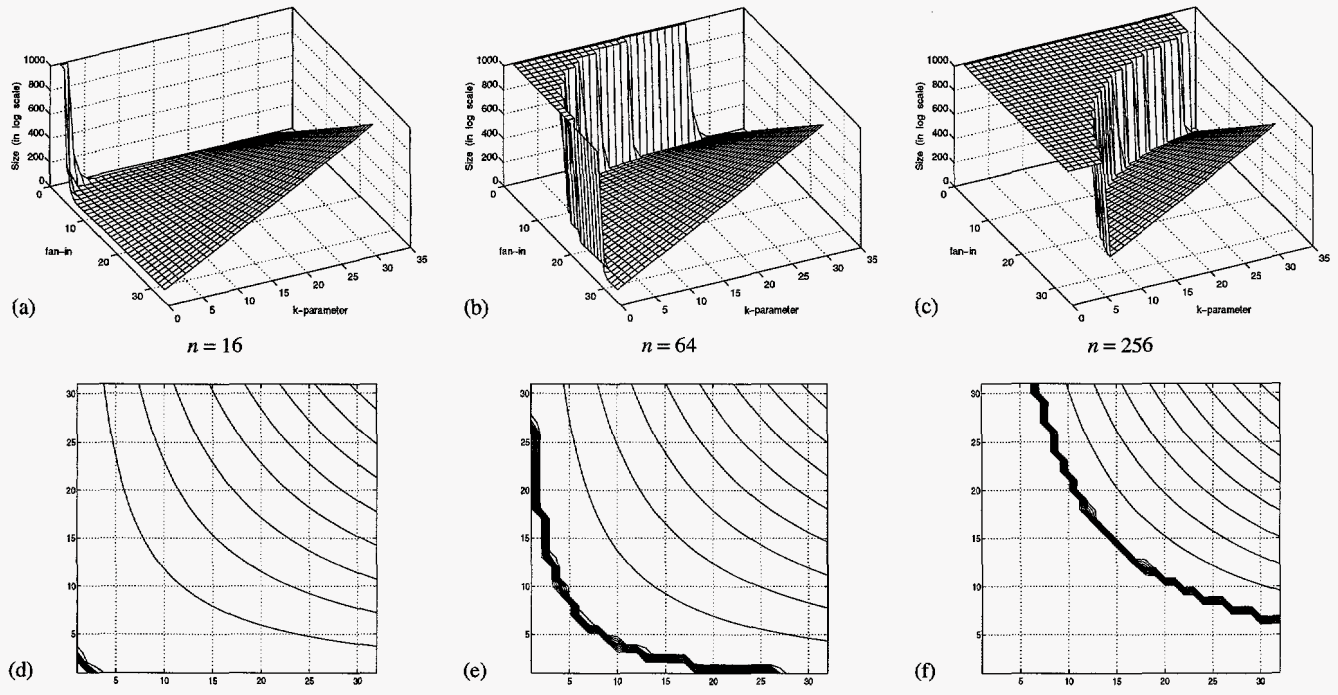

Fig. 2. The *size* (in logarithmic scale) of NNs implementing arbitrary BFs for: (a) $n = 16$; (b) $n = 64$; (c) $n = 256$ (clipped at 2^{1000}), and the contour plots for the same cases (d), (e), (f).

Proposition 5 (from [19]) The absolute minimum of size $_{BFs}$ is obtained for fan-in $\Delta = 2$.

Sketch of proof We will analyse only the critical points by using the approximation $k\Delta \approx n - \log n$. Intuitively the claim can be understood if we replace this value in (16):

$$\begin{aligned} size^*_{BFs} &\approx \mu \cdot 2^{n-\log n-k} + 2^{\Delta} \cdot 2^{2^{n-n+\log n-\Delta}} \\ &< \mu \cdot 2^{n-\log n} + 2^{\Delta} \cdot 2^{2^{\log n}} \\ &= \mu \cdot 2^n / n + 2^{\Delta} \cdot 2^n, \end{aligned}$$

which is minimised for $\Delta = 2$.

The detailed proof relies on computing $size_{BFs}$ as a function of (n, μ, k, Δ), for the critical points $k \approx (n - \log n)/\Delta$, and then showing that:

$$size^*_{BFs}(n, \mu, \Delta + 1) - size^*_{BFs}(n, \mu, \Delta) > 0, \qquad (19)$$

therefore, the function is monotonically increasing and the minimum is obtained for the smallest *fan-in* $\Delta = 2$. ❑

Remark It is to be mentioned that the other relative minima (on, or in the vicinity of the parabola $k\Delta \approx n - \log n$) might have more practical interest as leading to networks having fewer layers ($n/\log\Delta$ instead of n).

B. $IF_{n,m}$ Functions Using Threshold Gates

Similar optimal results can be obtained for implementing $IF_{n,m}$ functions. Beside the *size*-optimal solution detailed in [83], another solution was presented in [25, 26, 27], and later improved in [22, 23]. It has a first layer of COMPARISONs followed by a second layer of MAJORITY gates. Because the first layer is represented by COMPARISONs (*i.e.*, $IF_{n,1}$ functions), it is possible to decompose them such as to satisfy the limited *fan-in* condition [11–15, 23]. The previous known results were that COMPARISON:

- cannot be computed by a single TG with polynomially (in the number of inputs) bounded integer *weights*;
- can be computed by a *depth* = 2 NN with $\boldsymbol{O}(n^4)$ TGs and polynomially bounded *weights* [76];
- can be computed by a *depth* = 3 NN with $3n$ TGs and polynomially bounded *weights* [87];
- can be computed by a *depth* = 3 NN with $\Theta(n/\log n)$ TGs and polynomially bounded *weights* [84].
- can be computed by a *depth* = 2 linear threshold network of *size* $= 2\lceil n/\lceil\sqrt{n}\rceil\rceil$, with *weight* values of at most $2^{\lceil\sqrt{n}\rceil}$, and with an upper bound of $2\lceil\sqrt{n}\rceil + 1$ for the maximum *fan-in* [97].

The solution presented in [25] has evolved into a class of solutions [26, 27] which covers almost all the other solutions (for more details see [15]).

Proposition 6 (from [26]) The COMPARISON of two n-bit numbers (i.e., $IF_{n,1}$ functions) can be computed by a Δ-ary tree of size $\boldsymbol{O}(n/\Delta)$ and depth $\boldsymbol{O}(\log n/\log\Delta)$ for any $3 \le \Delta \le 2n$.

Proof Let $\boldsymbol{X} = x_{n-1}\, x_{n-2} \ldots x_1\, x_0$ and $\boldsymbol{Y} = y_{n-1}\, y_{n-2} \ldots y_1\, y_0$ be the two binary numbers (integers) of n bits each. The COMPARISON of the two numbers is defined as:

$$C_n^{>(\geq)}(\boldsymbol{X}, \boldsymbol{Y}) = \begin{cases} 1 & \text{if } \boldsymbol{X} > \boldsymbol{Y}\ (\boldsymbol{X} \geq \boldsymbol{Y}) \\ 0 & \text{if } \boldsymbol{X} \leq \boldsymbol{Y}\ (\boldsymbol{X} < \boldsymbol{Y}) \end{cases} \qquad (20)$$

where the subscript indicates the number of bits (length). We should mention that $C_n^{>}$ and $C_n^{\geq}$ are *isobaric* functions (*i.e.*, functions which can be implemented by TGs having identical *weights*, but different *thresholds* [50]), so there is no relevant difference:

$$\begin{aligned} C_n^{\geq}(\boldsymbol{X}, \boldsymbol{Y}) &= C_n^{>}(\boldsymbol{X}+1, \boldsymbol{Y}) \\ &= C_n^{>}(\boldsymbol{X}, \boldsymbol{Y}-1) \end{aligned} \qquad (21)$$

(when working with TGs the addition or subtraction of 1 can be done by changing the value of the *threshold*).

We shall describe the first layer of TGs and, then, use two recursive equations which make it possible to reduce one layer to the next layer (for the particular case $n = \Delta = 4$ the resulting architecture can be seen in in Fig. 3(a)).

Divide $\boldsymbol{X}$ in λ groups: $\Xi_{\lambda-1}, \Xi_{\lambda-2}, \ldots, \Xi_1, \Xi_0$ (these groups are not necessarily equal, but, for ease of notations, we shall consider them equal), where $\Xi_i = x_{(i+1)n/\lambda-1} \ldots x_{in/\lambda}$ (here we have also considered n divisible by λ, which is also not really necessary but simplifies notations). Similarly, divide $\boldsymbol{Y}$ in λ groups: $\Psi_{\lambda-1}, \Psi_{\lambda-2}, \ldots, \Psi_1, \Psi_0$, with $\Psi_i = y_{(i+1)n/\lambda-1} \ldots y_{in/\lambda}$. Now, take $\lambda = 2n/\Delta$ and build the first layer of TGs by comparing each group Ξ_i with the corresponding group Ψ_i using two *isobaric* TGs, one for $C^{>}_{\Delta/2}(\Xi_i, \Psi_i)$:

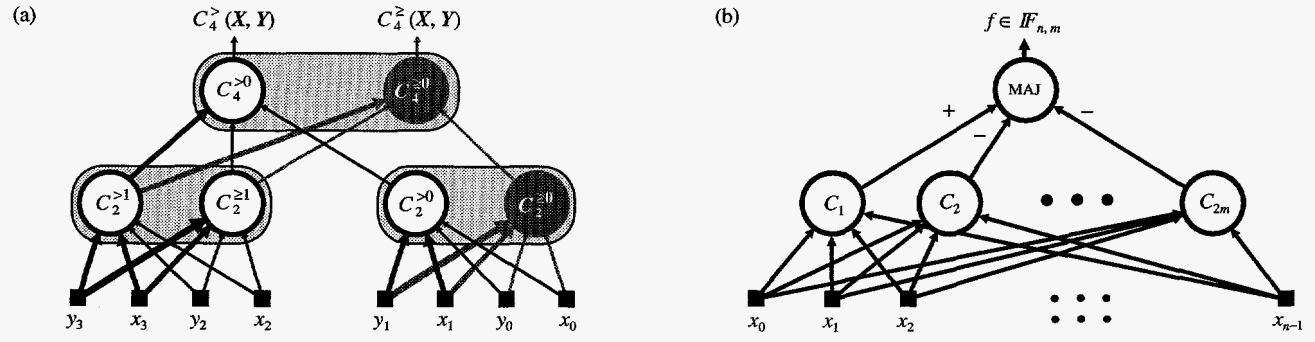

Fig. 3. (a) Tree decomposition of COMPARISON for $n = \Delta = 4$; TGs are represented by circles; nodes are light tinted; not used TGs are dark tinted; thin connections represent $|weight| = 1$; thick connections represent $|weight| = 2$. (b) Solution for implementing $IF_{n,m}$ functions using COMPARISONs and MAJORITY gate.

$$C^{>}_{\Delta/2}(\Xi_i, \Psi_i)$$

$$= C^{\geq}_{\Delta/2}(\Xi_i, \Psi_i + 1)$$

$$= sgn\left[\sum_{j=0}^{\Delta/2-1} 2^j (x_{i\Delta/2+j} - y_{i\Delta/2+j}) - 1\right] \quad (22)$$

and the other one for $C^{\geq}_{\Delta/2}(\Xi_i, \Psi_i)$:

$$C^{\geq}_{\Delta/2}(\Xi_i, \Psi_i) = sgn\left[\sum_{j=0}^{\Delta/2-1} 2^j (x_{i\Delta/2+j} - y_{i\Delta/2+j})\right]. \quad (23)$$

To avoid encumbering notations, we shall use $C^{>i}_{\Delta/2}$ instead of $C^{>}_{\Delta/2}(\Xi_i, \Psi_i)$, and $C^{\geq i}_{\Delta/2}$ instead of $C^{\geq}_{\Delta/2}(\Xi_i, \Psi_i)$. Here the subscripts show how many bits (from one number) are compared, and the superscripts represent the number (*i.e.*, an index) of the group.

Further, consider the outputs from the first layer as two numbers of $(2n/\Delta)$ bits each:

$$C^{>2n/\Delta-1}_{\Delta/2}, \ldots, C^{>0}_{\Delta/2} \text{ and } C^{\geq 2n/\Delta-1}_{\Delta/2}, \ldots, C^{\geq 0}_{\Delta/2}. \quad (24)$$

Suppose now that we are at the l-th level of the Δ-ary tree (l-th layer), and that the inputs are two k-bit numbers:

$$C_B^{>k-1}, \ldots, C_B^{>0} \text{ and } C_B^{\geq k-1}, \ldots, C_B^{\geq 0}. \quad (25)$$

The value $B = (\Delta/2)^l$ is obtained from the fact that at each level a reduction by $\Delta/2$ takes place. Divide these k-bit numbers in equal groups of $\Delta/2$ bits each. We have $2k/\Delta$ groups, and it is not difficult to see that level $l+1$ can be built out of $2k/\Delta$ nodes ($i = 0, 1, \ldots, 2k/\Delta - 1$). Each node has two circuits for computing $C^{>i}_{B\Delta/2}$ and $C^{\geq i}_{B\Delta/2}$. By using the notation $D = i(\Delta/2)$, we can write:

$$C^{>i}_{B\Delta/2} = \left\{\bigvee_{j=0}^{\Delta/2-2}\left[C_B^{>D+j} \wedge \left(\bigwedge_{k=j+1}^{\Delta/2-1} C_B^{\geq D+k}\right)\right]\right\} \vee C_B^{>D+\Delta/2-1} \quad (26)$$

and respectively:

$$C^{\geq i}_{B\Delta/2} = \left(\bigwedge_{j=0}^{\Delta/2-1} C_B^{\geq D+j}\right) \vee \left\{\bigvee_{j=1}^{\Delta/2-2}\left[C_B^{>D+j} \wedge \left(\bigwedge_{k=j+1}^{\Delta/2-1} C_B^{\geq D+k}\right)\right]\right\} \vee C_B^{>D+\Delta/2-1}. \quad (27)$$

Both these circuits have *fan-in* $= \Delta - 1$ as out of all the possible Δ inputs ($\Delta/2$ being $C_B^{>D+\Delta/2-1}, \ldots, C_B^{>D}$, and $\Delta/2$ being $C_B^{\geq D+\Delta/2-1}, \ldots, C_B^{\geq D}$) we do not use:

- $C_B^{\geq D}$ for computing $C^{>i}_{B\Delta/2}$ in (26);
- $C_B^{>D}$ for computing $C^{\geq i}_{B\Delta/2}$ in (27).

As the outputs of the first layer are two numbers representing results of partial COMPARISONs (as shown by (24)), we can apply (26) and (27) to build the second layer. Because the outputs of this second layer can also be interpreted as two numbers representing results of partial COMPARISONs, we can again (*i.e.*, recurrently) apply (26) and (27) to build all the subsequent layers.

The *depth* of the resulting tree can be computed from the fact that the last node (the root) has to cover all the n input bits, thus $B = (\Delta/2)^l$ becomes $n = (\Delta/2)^{depth_{\text{COMP}}}$ leading to:

$$depth_{\text{COMP}} = \lceil \log n / (\log\Delta - 1) \rceil = \boldsymbol{O}(\log n / \log\Delta). \quad (28)$$

The first layer has $n/(\Delta/2) = 2n/\Delta$ nodes of two threshold gates: $C^{>i}_{\Delta/2}$ and $C^{\geq i}_{\Delta/2}$ (as given by (22) and (23)). Each subsequent layer reduces the number of nodes by $\Delta/2$, and also has two circuits per node: $C^{>i}_{B\Delta/2}$ and $C^{\geq i}_{B\Delta/2}$ (as given by (26) and (27)). We are now able to compute the *size* as the following sum:

$$size_{\text{COMP}} = 2 \cdot \frac{n}{\Delta/2} + 2 \cdot \frac{n}{(\Delta/2)^2} + \ldots + 2 \cdot \frac{n}{(\Delta/2)^{depth_{\text{COMP}}}}. \quad (29)$$

By simple mathematics, this gives:

$$size_{\text{COMP}} = \lceil 4(n-1)/(\Delta-2) \rceil = \boldsymbol{O}(n/\Delta) \quad (30)$$

which concludes the proof. ❑

Remark I On each layer the rightmost node will need only one circuit: either $C^{>0}_{B\Delta/2}$ or $C^{\geq 0}_{B\Delta/2}$ (see Fig. 3(a)). The *size* given by (30) is, thus, reduced with one circuit per layer, and there are $depth_{\text{COMP}}$ layers (28). The exact value for *size* is:

$$size_{\text{COMP}} = \lceil 4(n-1)/(\Delta-2) \rceil - depth_{\text{COMP}} = \lceil 4(n-1)/(\Delta-2) \rceil - \lceil \log n / (\log\Delta - 1) \rceil. \quad (31)$$

Remark II The proof has been given for Δ even, but the decomposition also holds for Δ odd. If Δ is even, all the TGs from the first layer (leaves) have *fan-in* $= \Delta$ (there are $\Delta/2$ connections to bits of $\boldsymbol{X}$ and $\Delta/2$ connections to bits of $\boldsymbol{Y}$), *weights* $\leq 2^{\Delta/2-1}$, and *thresholds* $\in \{-1, 0\}$ (see (22) and (23)). All the other circuits from the nodes of the tree have *fan-in* $= \Delta - 1$. If Δ is odd, things are reversed: all the TGs from the first layer have *fan-in* $= \Delta - 1$, while all the other circuits have *fan-in* $= \Delta$. The resulting tree is an incomplete Δ-ary tree as anyhow some nodes will have $\Delta - 1$ connections, while others will have Δ.

The nodes from the decomposition tree are computing BFs f_Δ, which form a particular class of functions $\boldsymbol{IF}_\Delta$, namely *"the class of functions* $f_\Delta = f_\Delta(g_{\Delta/2-1}, e_{\Delta/2-1}, \ldots, g_0, e_0)$ *of* Δ *input variables, with* Δ *even, and computing:*

$$f_\Delta \stackrel{def}{=} \bigvee_{j=0}^{\Delta/2-1}\left[g_j \wedge \left(\bigwedge_{k=j+1}^{\Delta/2-1} e_k\right)\right]." \quad (32)$$

By convention, we consider $\bigwedge_{i=\alpha}^{\alpha-1} e_i \stackrel{def}{=} 1$. One restriction is that the input variables are pair-dependent, meaning that we

can group the Δ input variables in $\Delta/2$ pairs of two input variables each: $(g_{\Delta/2-1}, e_{\Delta/2-1}), \ldots, (g_0, e_0)$, and that in each such group one variable is 'dominant' (*i.e.*, when a dominant variable is 1, the other variable forming the pair will also be 1). Formally:

$$IF_\Delta \stackrel{def}{=} \{ f_\Delta \mid f_\Delta : \{(0,0),(0,1),(1,1)\}^{\Delta/2} \rightarrow \{0,1\},$$
$$\Delta/2 \in \mathbb{N}^*, f_\Delta \stackrel{def}{=} \bigvee_{j=0}^{\Delta/2-1} [g_j \wedge (\bigwedge_{k=j+1}^{\Delta/2-1} e_k)],$$
$$g_i \Rightarrow e_i,\ i = 0, 1, \ldots, \Delta/2-1 \} . \quad (33)$$

This is a class of linearly separable BFs, and it can be shown that the absolute *weights* and *thresholds* for implementing any function from this class are upper bounded by $2^{\Delta/2}$.

Proposition 7 (from [23]) IF_Δ is a class of linearly separable functions.

Proof The proof is constructive, and shows that, by copying the *weights* from f_Δ, adding two new *weights* and modifying the *threshold* we can find $f_{\Delta+2}$, such as a recursive version of (32) is satisfied. Thus, all $f_\Delta \in IF_\Delta$ can be implemented by one TG having *fan-in* $= \Delta - 1$.

For $\Delta = 4$, (32) becomes:

$$f_4(g_1, e_1, g_0, e_0) = g_1 \vee (e_1 \wedge g_0)$$

which is a linearly separable function:

$$g_1 \vee (e_1 \wedge g_0) = \langle 2g_1 + e_1 + g_0 \rangle_{1.5}$$
$$= sgn(2g_1 + e_1 + g_0 - 2). \quad (34)$$

Refining (32), we can determine the following recursive version (we increment by 2, as Δ has to be even):

$$f_{\Delta+2} = f_{\Delta+2}(g_{\Delta/2}, e_{\Delta/2}, \ldots, g_0, e_0)$$
$$= \bigvee_{j=0}^{\Delta/2} [g_j \wedge (\bigwedge_{k=j+1}^{\Delta/2} e_k)]$$
$$= \bigvee_{j=0}^{\Delta/2-1} \{ g_j \wedge [(\bigwedge_{k=j+1}^{\Delta/2-1} e_k) \wedge e_{\Delta/2}] \} \vee (g_{\Delta/2} \wedge 1)$$
$$= g_{\Delta/2} \vee [e_{\Delta/2} \wedge f_\Delta(g_{\Delta/2-1}, e_{\Delta/2-1}, \ldots, g_0, e_0)]$$
$$= g_{\Delta/2} \vee (e_{\Delta/2} \wedge f_\Delta). \quad (35)$$

Suppose now that the claim is true for Δ (*i.e.*, f_Δ is linearly separable), then:

$$f_\Delta = sgn(\sum_{i=0}^{\Delta/2-1} v_i g_i + \sum_{i=0}^{\Delta/2-1} w_i e_i + t_\Delta) \quad (36)$$

is true. As hypothesis for recursion, we shall also consider that all the *weights* are positive (non-negative) integers, while the *thresholds* are negative integers (easy to verify for the particular case $\Delta = 4$ by simply looking at (34): $v_1 = 2$, $w_1 = 1$, $v_0 = 1$, $w_0 = 0$, while $t_4 = -2$).

To constructively prove that $f_{\Delta+2}$ is linearly separable, we build it in three steps:

- *copy all the corresponding weights from f_Δ*;
- add two additional *weights* $v_{\Delta/2}$ and $w_{\Delta/2}$ (which correspond to the variables $g_{\Delta/2}$ and $e_{\Delta/2}$):

$$v_{\Delta/2} = 1 + \sum_{i=0}^{\Delta/2-1} w_i \quad (37)$$

$$w_{\Delta/2} = \sum_{i=0}^{\Delta/2-1} v_i ; \quad (38)$$

- change the *threshold* to $t_{\Delta+2}$:

$$t_{\Delta+2} = -1 - \sum_{i=0}^{\Delta/2-1} v_i - \sum_{i=0}^{\Delta/2-1} w_i . \quad (39)$$

Replacing (37), (38) and (39) in (36) we have:

$$f_{\Delta+2} = sgn[(v_{\Delta/2} g_{\Delta/2} + \sum_{i=0}^{\Delta/2-1} v_i g_i) + (w_{\Delta/2} e_{\Delta/2} + \sum_{i=0}^{\Delta/2-1} w_i e_i) + t_{\Delta+2}]. \quad (40)$$

We shall verify that (36) and (40) satisfy the recursion (35). Three cases have to be considered:

- If $g_{\Delta/2} = 1$, then $f_{\Delta+2} = 1$ regardless of the other input variables (see (35)). By hypothesis we also have $e_{\Delta/2} = 1$, hence (40) becomes:

$$f_{\Delta+2} = sgn[(v_{\Delta/2} + \sum_{i=0}^{\Delta/2-1} v_i g_i) + (w_{\Delta/2} + \sum_{i=0}^{\Delta/2-1} w_i e_i) + t_{\Delta+2}].$$

  The worst case — due to the fact that all the *weights* are positive — is when all the other input variables are 0. By substituting (37), (38) and (39) in the previous equation we obtain:

$$f_{\Delta+2} = sgn(v_{\Delta/2} + w_{\Delta/2} + t_{\Delta+2})$$
$$= sgn(0)$$
$$= 1.$$

If $g_{\Delta/2} = 0$ we have to analyse two cases.

- First, suppose that $e_{\Delta/2} = 0$. This makes $f_{\Delta+2} = 0$ regardless of the other input variables (see (35)). Now, (40) can be rewritten:

$$f_{\Delta+2} = sgn(\sum_{i=0}^{\Delta/2-1} v_i g_i + \sum_{i=0}^{\Delta/2-1} w_i e_i + t_{\Delta+2})$$

  and even if all the (other) input variables are 1, the value of the threshold $t_{\Delta+2}$ (see (39)) is large enough, such that $f_{\Delta+2} = sgn(-1) = 0$.
- The last — and most complicated case — is $e_{\Delta/2} = 1$ (we do remember that $g_{\Delta/2} = 0$). In this case, $f_{\Delta+2} = f_\Delta$ (see (35)). Starting again from (40), we have:

$$f_{\Delta+2} = sgn[\sum_{i=0}^{\Delta/2-1} v_i g_i + (w_{\Delta/2} + \sum_{i=0}^{\Delta/2-1} w_i e_i) + t_{\Delta+2}]$$

  and by substituting (38) and (39) we obtain:

$$f_{\Delta+2} = sgn[\sum_{i=0}^{\Delta/2-1} v_i g_i + (\sum_{i=0}^{\Delta/2-1} v_i + \sum_{i=0}^{\Delta/2-1} w_i e_i) - 1 - \sum_{i=0}^{\Delta/2-1} v_i - \sum_{i=0}^{\Delta/2-1} w_i]$$
$$= sgn[(\sum_{i=0}^{\Delta/2-1} v_i g_i + \sum_{i=0}^{\Delta/2-1} w_i e_i) - 1 - \sum_{i=0}^{\Delta/2-1} w_i]$$

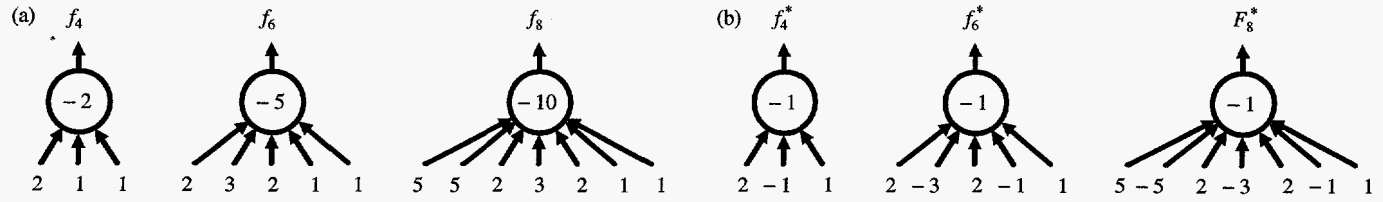

Fig. 4. (a) The series of *weights*: 1, 1, 2, 3, 2, 5, 5 and the corresponding *thresholds*: $-2, -5, -10$ for f_4, f_6, f_8. (b) The series of alternate signs *weights* (in our proof all the *weights* are positive): $1, -1, 2, -3, 2, -5, 5$ leading to constant *threshold* (-1), for f_4^*, f_6^*, f_8^*; these BFs are identical with f_4, f_6, f_8 if the e_i inputs are inverted, *i.e.*, $f_\Delta^* = f_\Delta(g_{\Delta/2-1}, \bar{e}_{\Delta/2-1}, \ldots, g_0, \bar{e}_0)$.

The first two sums are larger (or, smaller) than $-t_\Delta$ (see (36)) if $f_\Delta = 1$ (or, respectively $f_\Delta = 0$). Let these two sums be $-t_\Delta + \varepsilon$, with $\varepsilon > 0$ if $f_\Delta = 1$, and respectively $\varepsilon < 0$ if $f_\Delta = 0$. Then:

$$f_{\Delta+2} = sgn\left[(-t_\Delta + \varepsilon) - 1 - \sum_{i=0}^{\Delta/2-1} w_i\right]$$

and replacing t_Δ as given by (39):

$$f_{\Delta+2} = sgn\left\{\left[\left(1 + \sum_{i=0}^{\Delta/2-2} v_i + \sum_{i=0}^{\Delta/2-2} w_i\right) + \varepsilon\right] - 1 - \sum_{i=0}^{\Delta/2-1} w_i\right\}$$

$$= sgn\left(\sum_{i=0}^{\Delta/2-2} v_i + \varepsilon - w_{\Delta/2-1}\right)$$

Finally, we use (38) to obtain:

$$f_{\Delta+2} = sgn\left(\sum_{i=0}^{\Delta/2-2} v_i + \varepsilon - \sum_{i=0}^{\Delta/2-2} v_i\right)$$

$$= sgn(\varepsilon)$$

$$= f_\Delta.$$

The fact that the recursion (35) is verified concludes the proof. ❑

Remark It should be mentioned that the operation of copying the *weights* of a linear separable function to build another one — and possibly changing the *threshold* — does not give rise to a linearly separable function in general. This has been proven by Walker *et al.* [99] where the conclusion followed that: *"no direct mapping of weights exists between fully and limited-interconnect nets." Proposition 7* shows that there are particular classes of BFs which have the property that such a direct mapping of *weights* exists and can be found.

Another interesting property of this class of functions is that the *weights* and *thresholds* (of $\mathbb{F}_\Delta$ functions) are bounded.

Proposition 8 (from [23]) The absolute weights and thresholds of the gates implementing $\forall f_\Delta \in \mathbb{F}_\Delta$ *are bounded by* $2^{\Delta/2}$.

Proof By solving the system of recurrent equations (37) and (38) with the initial conditions $v_1 = 2$, $w_1 = 1$, $v_0 = 1$, $w_0 = 0$ (see (34)), we found that for $\forall i \geq 4$ (*i.e.*, $\Delta \geq 8$):

$$v_i = w_i = 5 \cdot 2^{i-3}.$$

By replacing i with the largest possible value ($i = \Delta/2 - 1$), we get the maximal *weights*:

$$v_{\Delta/2-1} = w_{\Delta/2-1} = 5 \cdot 2^{\Delta/2-4} = 5/16 \cdot 2^{\Delta/2} < 2^{\Delta/2}.$$

From (37), (38) and (39) we know that $t_\Delta = -v_{\Delta/2-1} - w_{\Delta/2-1}$, so the absolute maximum *threshold* (for $\Delta \geq 8$) is:

$$|t_\Delta| = 5 \cdot 2^{\Delta/2-3} = 5/8 \cdot 2^{\Delta/2} < 2^{\Delta/2}$$

concluding the proof. ❑

The TGs which realise the first three $\mathbb{F}_\Delta$ functions (f_4, f_6, f_8) can be seen in Fig. 4(a), while in Fig. 4(b) a constant *threshold* solution $f_\Delta^* = f_\Delta(g_{\Delta/2-1}, \bar{e}_{\Delta/2-1}, \ldots, g_0, \bar{e}_0)$ is presented.

Proposition 9 The COMPARISON *of two n-bit numbers can be computed by a neural network having* $\boldsymbol{O}(n/\Delta)$ *size and* $\boldsymbol{O}(\log n / \log\Delta)$ *depth. The integer weights and thresholds are: (i) polynomially bounded for all the values of the fan-in in the range* $3 \leq \Delta \leq c \log n$*; (ii) super-polynomially bounded for all the values of the fan-in in the range* $c \log n < \Delta \leq c \log^k n$*; and (iii) exponentially bounded for all the values of the fan-in in the range* $c \log^k n < \Delta \leq 2n$.

Proof We use *Proposition 6* to build the Δ-ary tree. From *Proposition 7* we know that all the functions are linearly separable. Finally, using *Proposition 8*, we shall prove the bounded *weights* condition.

In *Proposition 6* we have shown how to build a Δ-ary tree of *size* $\boldsymbol{O}(n/\Delta)$ and *depth* $\boldsymbol{O}(\log n / \log\Delta)$ to compute COMPARISON. As only the leaves (of the tree) are TGs, we use *Proposition 7* which shows that each of the circuits from the nodes of the subsequent layers can be implemented by one TG. Clearly, $C_{B\Delta/2}^{>i}$ is an f_Δ function (see (26) and (32)), so it can be realised by a TG. $C_{B\Delta/2}^{\geq i}$ is also an f_Δ function either by rewriting (27) such as to look like (32), or by using (21) which shows that $C_{B\Delta/2}^{\geq i}$ is TG realisable as $C_{B\Delta/2}^{>i}$ is.

For odd values of Δ, the decomposition tree can be built in the following way:

- the first layer is made of $\Delta - 1$ *fan-in* TGs ($\Delta - 1$ being now even);
- all the other layers implement $f_{\Delta+1}$ functions — but any f_Δ function has $\Delta - 1$ inputs (see (32) and Fig. 3(a)) — which implies that $f_{\Delta+1}$ functions have *fan-in* $= \Delta$ (as $(\Delta + 1) - 1 = \Delta$).

Lastly, the bounding conditions for *weights* and *thresholds* will be proven. By construction, all the *weights* and the *thresh-*

olds are integers. From *Proposition 6* we know that the *weights* of the TGs from the first layer are bounded by $2^{\Delta/2-1}$ and that the *thresholds* are either 0 or -1 (see (22) and (23)). From *Proposition 8* we know that the *weights* and the *thresholds* of all the other TGs are upper bounded by $2^{\Delta/2}$. These show that the bounds on the *weights* and *thresholds* are: (*i*) polynomial for $\Delta \le c \log n$; (*ii*) super-polynomial for $c \log n < \Delta \le c \log^k n$; (*iii*) exponential for $c \log^k n < \Delta \le 2n$; and concludes the proof. □

It is thus clear how one can lower the *fan-in* to a constant (*i.e.*, $\boldsymbol{O}(1)$) and obtain a linear *size* NN having logarithmic *depth*. A more interesting result is to let *fan-in* $= \boldsymbol{O}(\log n)$; now, the very slow growth of *depth* $\boldsymbol{O}(\log n / \log\log n)$, makes it 'almost constant' for normal values of n: *depth* ≤ 7 for $n \le 10^6$. This increase of the *fan-ins* is rewarded by a significant decrease of the *size* from linear to $\boldsymbol{O}(n/\log n)$. In the meantime, the *weights* grow only from constant to linear.

Proposition 10 Any function $f \in \mathbb{F}_{n,m}$ *can be computed by a neural network with polynomially bounded integer weights (and thresholds) having depth* $\boldsymbol{O}(\log(m n)/\log\Delta)$ *and size* $\boldsymbol{O}(mn/\Delta)$ *for all the values of the fan-in in the range 3 to* $c \log n$.

The proof follows from *Proposition 9* [20, 27]. The *size* is increased $2m$ times, as $2m$ COMPARISONs are used in the first layer (see Fig. 3(b)). The *size* of the decomposition tree for the MAJORITY function (*i.e.*, function which can be implemented by a TG having all the *weights* ± 1) from the second layer is only $(2m-1)/(\Delta-1)$, and can be neglected [27], thus:

$$size_{\mathbb{F}} = 2nm \cdot \left\{ \frac{1}{\Delta/2} + \ldots + \frac{1}{(\Delta/2)^{depth_{\mathbb{F}}}} \right\}, \quad (41)$$

where $depth_{\mathbb{F}} = \lceil \log n / (\log\Delta - 1) \rceil$. The *depth* grows with the *depth* of the decomposition tree having $2m$ inputs implementing the MAJORITY function, which is $(\log m + 1)/\log\Delta$. A substantial *size* reduction is obtained if the *fan-in* is limited. Due to that, the maximum number of *different BFs* which can be computed in each layer is:

$$\frac{2n}{\Delta} 2^{\Delta}, \frac{2n/\Delta}{\Delta/2} 2^{\Delta(\Delta/2)}, \ldots, \frac{2n/\Delta}{(\Delta/2)^{depth_{\mathbb{F}}-1}} 2^{\Delta(\Delta/2)^{depth_{\mathbb{F}}-1}}. \quad (42)$$

For m large enough (needed for achieving a certain precision [14, 29, 100]), and/or n large enough, the first terms of the sum from (41) will be larger than the equivalent ones from (42). This is equivalent to the trick from [45], as the lower levels will compute *all the possible functions* using only limited *fan-in* COMPARISONs. Hence, the optimum *size* becomes:

$$size^*_{\mathbb{F}} = 2n \cdot \left\{ \sum_{i=1}^{k} \frac{2^{\Delta(\Delta/2)^{i-1}}}{\Delta(\Delta/2)^{i-1}} + \sum_{i=k+1}^{depth_{\mathbb{F}}} \frac{m}{(\Delta/2)^{i}} \right\} \quad (43)$$

explained as the same BFs are computed redundantly. In terms of *fan-in*, several exponentially decreasing terms will be replaced by double exponential increasing terms.

To get a better understanding we have done extensive simulations by considering that $m = 2^{\varepsilon n}$. Some of the results of these simulations can be seen in Fig. 5. They show that it is always possible to obtain a significant reduction of the *size* by properly choosing a small constant *fan-in*. It is to be mentioned that the *size* reduction is by a huge factor which is of the form $2^{\varepsilon n - c}$ for very small *fan-ins* $\Delta_{optim} = 4 \ldots 6$. Following similar steps to the ones used in *Proposition 5*, it is possible to show that the minimum *size* is obtained for very small *fan-ins* ($\Delta_{optim} = 3$).

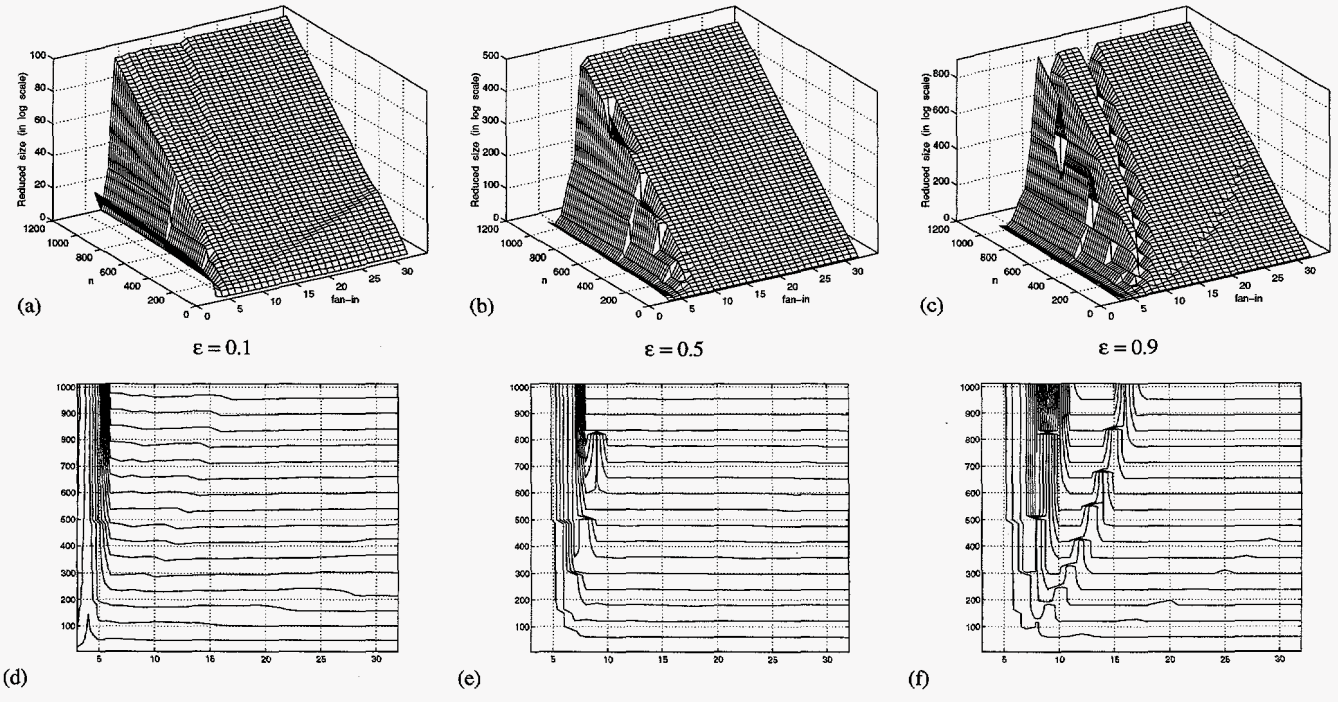

Fig. 5. The reduced *size* (in logarithmic scale) of NNs implementing $\mathbb{F}_{n,m}$ functions for $m = 2^{\varepsilon n}$: (a) $\varepsilon = 0.1$; (b) $\varepsilon = 0.5$; (c) $\varepsilon = 0.9$; and the contour plots for the same cases (d), (e), (f); the lowest values are obtained for very small constant *fan-in* values.

The method can be directly used for implementing k functions of m examples when the *fan-in* is limited by Δ.

Proposition 11 Any finite set of k functions f from $\mathbb{F}_{n,i}$ defined by m examples ($i \le m \le ik$), can be computed by a neural network with polynomially bounded integer weights and thresholds, of $\boldsymbol{O}\{m(2n+k)/\Delta\}$ size and $\boldsymbol{O}\{\log(mn)/\log\Delta\}$ depth for all the values of the fan-in in the range 3 to $\boldsymbol{O}(\log n)$.

The proof follows from *Proposition 10*. Because all the k function are using the same n inputs, the first layer will need $2m$ COMPARISONs in the worst case (for all the k functions). The *size* is obtained by adding the *size* of the k decomposition trees of $2m$ inputs (implementing the k MAJORITY functions) to the *size* of the $2m$ COMPARISONs.

Finally, we will show that even VLSI-optimal implementations of $\mathbb{F}_{n,m}$ functions are obtained for small constant *fan-ins* ($\Delta = 6\ldots9$). This result builds on the closer estimates of *area* and *delay* suggested in INTRODUCTION. Different estimates for *area* and delay have already been computed for $\mathbb{F}_{n,m}$ functions [15, 26]. Not wanting to complicate the proof, we shall determine the VLSI-optimal *fan-in* when implementing COMPARISON (the same result is valid for $\mathbb{F}_{n,m}$ functions as can be intuitively expected as: the delay is determined by the first layer of COMPARISONs, while the *area* is mostly influenced by the same first layer of COMPARISONs). We have chosen the following approximations: $area = \Sigma_{NN}(\Sigma_i |w_i| + |\theta|)$, and *depth* for delay, but other estimates lead to the same results [15] (the optimal AT^2 being $\boldsymbol{O}(n \log^2 n)$).

Proposition 12 (from [11, 15]) The VLSI-optimal neural network which computes the COMPARISON of two n-bit numbers has small-constant fan-in threshold gates, with small-constant bounded weights and thresholds.

Proof From *Proposition 6, 7,* and *8*, the AT^2 of COMPARISON can be determined (for details see [11, 15]):

$$AT^2 \equiv \frac{2^{\Delta/2}}{\Delta} \cdot \frac{8n\Delta - 6n - 5\Delta}{\Delta - 2} \cdot \left(\frac{\log n}{\log\Delta}\right)^2 = \boldsymbol{O}\{n\log^2 n \cdot 2^{\Delta/2}/(\Delta\log^2\Delta)\} \quad (44)$$

and we can compute the derivative:

$$\begin{aligned}\frac{d(AT^2)}{d\Delta} = \frac{2^{\Delta/2}\log^2 n}{\Delta^2(\Delta-2)^2\log^3\Delta} \times \Big(&8n\Delta^3\log\Delta - 22n\Delta^2\log\Delta \\ &+ 12n\Delta\log\Delta - 5\Delta^3\log\Delta + 10\Delta^2\log\Delta \\ &- \frac{16}{\ln 2}n\Delta^2\log\Delta + \frac{24}{\ln 2}n\Delta\log\Delta - \frac{24}{\ln 2}n\log\Delta \\ &+ \frac{10}{\ln 2}\Delta^2\log\Delta - \frac{32}{\ln 2}n\Delta^2 + \frac{88}{\ln 2}n\Delta - \frac{48}{\ln 2}n \\ &+ \frac{20}{\ln 2}\Delta^2 - \frac{40}{\ln 2}\Delta\Big).\end{aligned}$$

This — unfortunately — involves transcendental functions of the variables in an essentially non-algebraic way. By considering the simplified 'complexity' version (44) we obtain:

$$d(AT^2)/d\Delta \equiv d\{n\log^2 n \cdot 2^{\Delta/2}/(\Delta\log^2\Delta)\}/d\Delta = \frac{2^{\Delta/2}}{\Delta\log^2\Delta} \cdot \left(\frac{\ln 2}{2} - \frac{1}{\Delta} - \frac{2}{\Delta\ln\Delta}\right),$$

which, when equated to zero, leads to $\ln\Delta(\Delta\ln 2 - 2) = 4$ (again a transcendental equation). This has $\Delta_{optim} = 6$ as integer solution. because the *weights* and the *thresholds* are bounded by $2^{\Delta/2}$ (*Proposition 8*), the proof is concluded. ❑

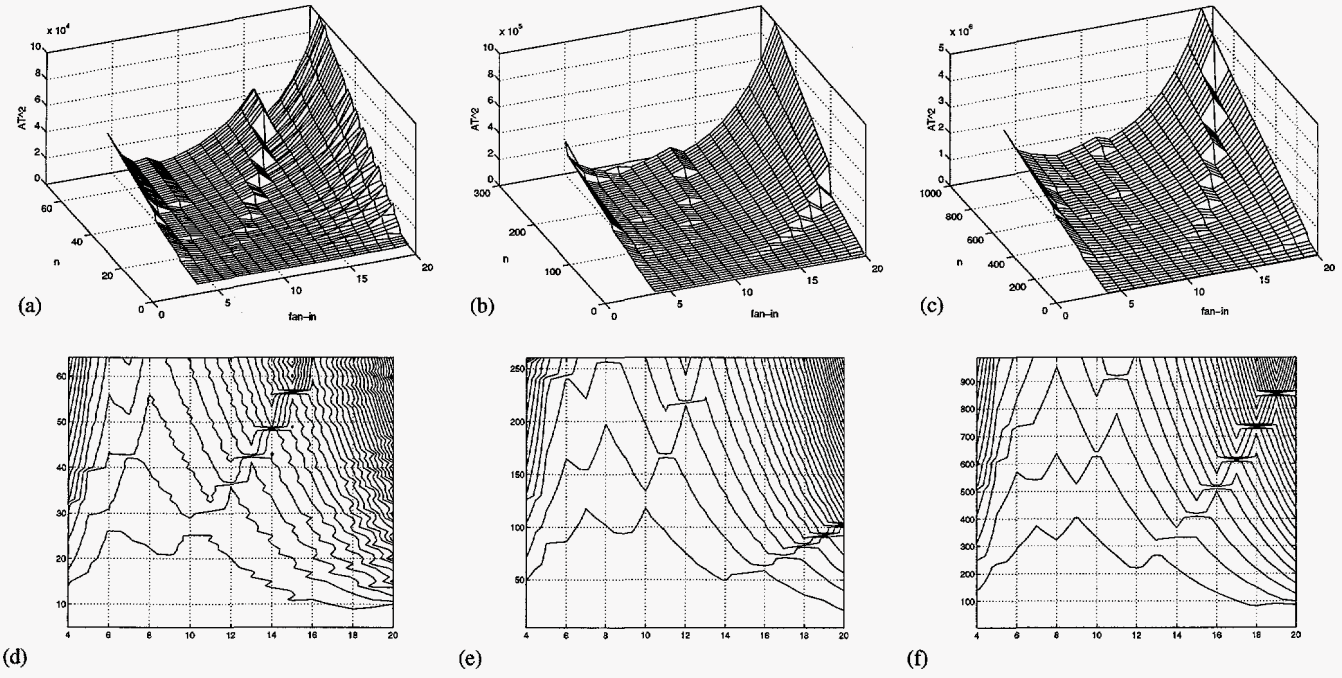

Fig. 6. The AT^2 values of COMPARISON — plotted as a 3D surface — versus the number of inputs n and the *fan-in* Δ for $4 \le \Delta \le 20$: (a) $n \le 64$; (b) $n \le 256$; (c) $n \le 1024$; (d), (e), and (f) show the contour plots for the same cases. Clearly the a 'valley' is formed, and the 'deepest' points constantly lie somewhere between $\Delta_{minim} = 6$ and $\Delta_{maxim} = 9$.

The minimum AT^2 is obtained for $\Delta_{optim} = 6...9$ (the proof has been obtained using approximations: neglecting ceilings, using the complexity estimate, etc.), as can be seen from the simulations (for variable *fan-ins* and different number of inputs n) presented in Fig. 6. We mention that there are similar *small constants* relating to our *capacity of processing information* [70]. This result has been extended to $IF_{n,m}$ functions. By extending this result to a three dimensional hardware implementation, the energy (VT^2 in this case) is minimised for *fan-ins* in the range $\Delta = 36...81$ (which are still small as opposed to the *fan-in* of the cells in the human brain, normally in the range $10^2...10^4$).

C. Boolean Functions Using Analog Neurons

A different approach is to use Kolmogorov's superpositions, which shows that there are NNs having only $2n+1$ neurons (*i.e.*, *size*-optimal) which can approximate any function. We start from a constructive solution for the general case [92–94].

Proposition 13 (from [92]) Define the function $\psi : \mathcal{E} \to \mathcal{D}$ *such that for each integer* $k \in N$:

$$\psi\left(\sum_{r=1}^{k} i_r \gamma^{-r}\right) = \sum_{r=1}^{k} \tilde{i}_r 2^{-m_r} \gamma^{-\frac{n^{r-m_{r-1}}}{n-1}} \tag{45}$$

where $\tilde{i}_r = i_r - (\gamma - 2)\langle i_r \rangle$ *and*

$$m_r = \langle i_r \rangle \times \{1 + \sum_{s=1}^{r-1} [i_s] \times ... \times [i_{r-1}]\}$$

for $r = 1, 2, ..., k$.

Here $\gamma \geq 2n+2$ is a base, $\mathcal{E} = [0, 1]$ is the unit interval, $\mathcal{D}$ is the set of terminating rational numbers $d_k = \sum_{r=1}^{k} i_r \gamma^{-r}$ defined on $k \in N$ digits ($0 \leq i_r \leq \gamma - 1$). Also, $\langle i_1 \rangle = [i_1] = 0$, while for $r \geq 2$: $\langle i_r \rangle = 0$ when $i_r = 0, 1, ..., \gamma - 2$, $\langle i_r \rangle = 1$ when $i_r = \gamma - 1$, $[i_r] = 0$ when $i_r = 0, 1, ..., \gamma - 3$, and $[i_r] = 1$ when $i_r = \gamma - 2, \gamma - 1$.

If we limit the functions to BFs, one digit ($k = 1$) is enough, which gives $\psi(0.i_1) = 0.i_1$, *i.e.* the identity function $\psi(x) = x$. Such a solution builds simple analog neurons having *fan-in* $\Delta \leq 2n+1$.

The known *weight* bounds (holding for $\Delta \geq 4$) are [72, 76, 82, 88]:

$$2^{(\Delta-1)/2} < weight < (\Delta+1)^{(\Delta+1)/2}/2^{\Delta}.$$

Thus, a *precision* of between Δ, and $\Delta \log \Delta$ bits per *weight* would be expected. Unfortunately, the constructive solution for Kolmogorov's superpositions requires a double exponential precision for ψ (45), and for the *weights*:

$$\alpha_p = \sum_{r=1}^{\infty} \gamma^{-(p-1)\frac{n^r-1}{n-1}}.$$

For BFs this precision is reduced to $(2n+2)^{-n}$, or $2n\log n$ bits per *weight*. Analog implementations are limited to just several bits of precision [60], this being one of the reasons for investigations on precision [33, 44, 95, 100], and on algorithms relying on limited integer *weights* [14, 34, 57]. Due to the limitation on precision the solution for implementing BFs should decompose the given BF in simpler BFs which can be efficiently implemented based on Kolmogorov's superpositions (*i.e.*, we have to reduce n to reasonable small values). The partial results from this first layer of analog building blocks can be combined using (again) Kolmogorov's superpositions, TGs or Boolean gates. The final mixed analog/digital implementation will requires more than three layers. It follows that a systematic solution which would utilise silicon to the best advantage would be to rewrite a given computation (*i.e.*, set of BFs) in a base larger than 2, and use Kolmogorov's superpositions for the analog implementation of the digit-wise computations in this larger base.

IV. Conclusions

The main conclusion of this overview paper is that hardware implementations of NNs are highly limited by the two dimensional mapping into silicon, which leads to limited *fan-in* and precision. For example, arbitrary BFs can be implemented using:

- classical Boolean gates, but require exponential *size*;
- TGs, but (again) in exponential *size* (still, there is an exponential gap in between);
- analog building blocks in linear *size* (having linear *fan-in* and polynomial precision *weights* and *thresholds*), the nonlinear activation function being the identity function.

Clearly, there are interesting *fan-in* dependent *depth-size* and *area-delay* tradeoffs, as well as optimal solutions having small constant *fan-in* values, and the problems are not alleviated by futuristic three dimensional optical implementations.

These results also suggest that:

- the brain does not optimise energy and power — like engineers do when designing integrated circuits — and might trade-off the slower individual speeds of its elementary computing elements (thus, reducing power), for their higher connectivity (larger *fan-ins*);
- two dimensional silicon implementations are limited with respect to connectivity, and might only slightly compensate by using higher computing speeds (see Fig. 1(a));
- three dimensional hardware implementations (*e.g.*, optical) might be still lagging behind biological ones with respect to connectivity, but it is to be expect that the higher computing speed might eventually compensate for that.

Future work should concentrate on finding closer estimates (*i.e.*, new cost functions) for comparing analog/digital as well as optical implementations.

References

[1] Y.S. Abu-Mostafa, "Connectivity versus entropy", in D.Z. Anderson (ed.), *Neural Information Processing Systems*, New York, NY: AIPress, 1988, pp. 1-8.

[2] M. Arai, "Bounds on the number of hidden units in binary-valued three-layer neural networks", *Neural Networks*, **6**, pp. 855-860, 1993.

[3] M.A. Arbib, *The Handbook of Brain Theory and Neural Networks*, Cambridge, MA: MIT Press, 1995.

[4] J.-G. Attali & G. Pagès, "Approximations of functions by a multilayer perceptron: a new approach", *Neural Networks*, **10**, pp. 1069-1081, 1997.

[5] A.R. Barron, "Universal approximation bounds for superpositions of a sigmoidal function", *IEEE Trans. Info. Theory*, **39**, pp. 930-945, 1993.

[6] P.L. Bartlett, "The sample complexity of pattern classification with neural networks: the size of the weights is more important than the size of the network", *Tech. Rep.*, Dept. Sys. Eng., Australian Natl. Univ., Canberra, 1996. Short version: "For valid generalization, the size of the weights is more important than the size of the network", in M.C. Mozer, M.I. Jordan & T. Petsche (eds.), *Advances in Neural Information Processing Systems*, Cambridge, MA: MIT Press, 1997, pp. 134-140.

[7] E.B. Baum, "On the capabilities of multilayer perceptrons", *J. Complexity*, **4**, pp. 193-215, 1988.

[8] V. Beiu, "Entropy bounds for classification algorithms", *Neural Network World*, **6**, pp. 497-505, 1996.

[9] V. Beiu, "Optimal VLSI Implementation of Neural Networks," in J.G. Taylor (ed.), *Neural Networks and Their Applications*, Chichester, UK: John Wiley, 1996, pp. 255-276.

[10] V. Beiu, "Digital integrated circuit implementations", in E. Fiesler & R. Beale (eds.), *Handbook of Neural Computation*, New York, NY: Oxford Univ. Press & Inst. of Physics, 1996, Chapter E1.4.

[11] V. Beiu, "Constant fan-in digital neural networks are VLSI-optimal", in S.W. Ellacott, J.C. Mason & I.J. Anderson (eds.), *Mathematics of Neural Networks: Models, Algorithms and Applications*, Boston, MA: Kluwer Academic, 1997, ch. 12, pp. 89-94.

[12] V. Beiu, "When constants are important", in I. Dumitrache (ed.), *Proc. Intl. Conf. on Control System & Computer Science CSCS-11*, Bucharest, Romania: UPB Press, 1997, vol. **2**, pp. 106-111.

[13] V. Beiu, "Optimization of circuits using a constructive learning algorithm", in A.B. Bulsari and S. Kallio (eds.), *Neural Networks in Engineering Systems*, Stockholm, Sweden: Systeemitekniikan seura ry (Systems Engineering Association, SEA) & Turku, Finland: Åbo Akademis Tryckeri, 1997, pp. 291-294.

[14] V. Beiu, "Reduced complexity constructive learning algorithm", *Neural Network World*, **8**, pp. 1-38, 1998.

[15] V. Beiu, "On the circuit and VLSI complexity of threshold gate COMPARISON", *Neurocomputing*, **19**, pp. 77-98, 1998.

[16] V. Beiu, *VLSI Complexity of Discrete Neural Networks*, Newark, NJ: Gordon & Breach, 1998/9 (to appear).

[17] V. Beiu & T. De Pauw, "Tight bounds on the size of neural networks for classification problems", in J. Mira, R. Moreno-Díaz & J. Cabestany (eds.), *Biological and Artificial Computation*, Berlin, Germany: Springer-Verlag, 1997, pp. 743-752.

[18] V. Beiu & S. Drăghici, "Limited weights neural networks: very tight entropy based bounds", in D.W. Pearson (ed.), *Proc. Intl. ICSC Symp. on Soft Computing SOCO'97*, Millet, Canada: ICSC Acad. Press, 1997, pp. 111-118.

[19] V. Beiu & H.E. Makaruk, "Deeper sparser nets can be optimal", *Neural Processing Letters*, **8**, 1998 (to appear).

[20] V. Beiu & J.G. Taylor, "VLSI optimal neural network learning algorithm", in D.W. Pearson, N.C. Steele & R.F. Albrecht (eds.), *Artificial Neural Nets and Genetic Algorithms*, New York, NY: Springer-Verlag, 1995, pp. 61-64.

[21] V. Beiu & J.G. Taylor, "Optimal mapping of neural networks onto FPGAs", in J. Mira and F. Sandoval (eds.), *From Natural to Artificial Neural Computation*, Berlin, Germany: Springer-Verlag, 1995, LNCS Vol. **930**, pp. 822-829.

[22] V. Beiu & J.G. Taylor, "Direct synthesis of neural networks", in *Proc. Intl. Conf. on Microelectronics for Neural Networks MicroNeuro'96*, Los Alamitos, CA: IEEE CS Press, 1996, pp. 257-264.

[23] V. Beiu & J.G. Taylor, "On the circuit complexity of sigmoid feedforward neural networks", *Neural Networks*, **9**, pp. 1155-1171, 1996.

[24] V. Beiu, S. Drăghici & T. De Pauw, "A constructive approach to calculating lower entropy bounds", *Neural Processing Letters*, **8**, 1998 (to appear).

[25] V. Beiu, J.A. Peperstraete, J. Vandewalle & R. Lauwereins, "Efficient Decomposition of COMPARISON and Its Applications," in M. Verleysen (ed.), *European Symp. on Artif. Neural Networks ESANN'93*, Brussels, Belgium: Dfacto, 1993, pp. 45-50.

[26] V. Beiu, J.A. Peperstraete, J. Vandewalle & R. Lauwereins, "Area-time performances of some neural computations", in P. Borne, T. Fukuda & S.G. Tzafestas (eds.), *Proc. Intl. Symp. on Signal Processing, Robotics, and Neural Networks SPRANN'94*, Lille, France: GERF EC, 1994, pp. 664-668.

[27] V. Beiu, J.A. Peperstraete, J. Vandewalle & R. Lauwereins, "Learning from Examples and VLSI Implementation of Neural Networks," in R. Trappl (ed.), *Cybernetics and Sys. Research EMCSR'94*, Singapore: World Scientific, 1994, pp. 1767-1774.

[28] E. Blum & K. Li, "Approximation theory and feedforward networks", *Neural Networks*, **4**, pp. 511-515, 1991.

[29] J. Bruck & J.W. Goodmann, "On the power of neural networks for solving hard problems", in D.Z. Anderson (ed.), *Neural Information Processing Systems*, New York, NY: AIPress, 1988, pp. 137-143. Also in *J. Complexity*, **6**, pp. 129-135, 1990.

[30] A. Bulsari, "Some analytical solutions to the general approximation problem for feedforward neural networks", *Neural Networks*, **6**, pp. 991-996, 1993.

[31] G. Cybenko, "Continuous valued neural networks with two hidden layers are sufficient", *Tech. Rep.*, Maths. Dept., Tufts Univ., Medford, 1988.

[32] G. Cybenko, "Approximations by superpositions of a sigmoid function", *Math. of Control, Signals & Systems*, **2**, pp. 303-314, 1989.

[33] J.S. Denker & B.S. Wittner, "Network generality, training required, and precision required", in D.Z. Anderson (ed.), *Neural Information Processing Systems*, New York, NY: AIPress, 1988, pp. 219-222.

[34] S. Drăghici & I.K. Sethi, "On the possibilities of the limited precision weights neural networks in classification problems", in J. Mira, R. Moreno-Díaz & J. Cabestany (eds.), *Biological and Artificial Computation*, Berlin, Germany: Springer-Verlag, 1997, pp. 753-762.

[35] E. Fiesler & R. Beale, *Handbook of Neural Computation*, New York, NY: Oxford Univ. Press & Inst. of Physics, 1996.

[36] K.-I. Funahashi, "On the approximate realization of continuous mapping by neural networks", *Neural Networks*, **2**, pp. 183-192, 1989.

[37] K.-I. Funahashi & Y. Nakamura, "Approximation of dynamical systems by continuous time recurrent neural networks", *Neural Networks*, **6**, pp. 801-806, 1993.

[38] S. Geva & J. Sitte, "A constructive method for multivariate function approximation by multilayered perceptrons", *IEEE Trans. Neural Networks*, **3**, pp. 621-623, 1992.

[39] M. Glesner & W. Pöchmüller, *Neurocomputers – An Overview of Neural Networks in VLSI*, London, UK: Chapman & Hall, 1994.

[40] D. Hammerstrom, "The connectivity analysis of simple association – or– how many connections do you need", in D.Z. Anderson (ed.), *Neural Information Processing Systems*, New York, NY: AIPress, 1988, pp. 338-347.

[41] E. Hartman, J.D. Keeler & J.M. Kowalski, "Layered neural networks with gaussian hidden units as universal approximations", *Neural Computation*, **2**, pp. 210-215, 1989.

[42] M.H. Hassoun, *Fundamentals of Artificial Neural Networks*. Cambridge, MA: MIT Press, 1995.

[43] R. Hecht-Nielsen, "Kolmogorov's mapping neural network existence theorem", in *Proc. IEEE Intl. Conf. on Neural Networks ICNN'87*, New York, NY: IEEE CS Press, 1987, vol. **3**, pp. 11-14.

[44] J.L. Holt & J.-N. Hwang, "Finite precision error analysis of neural network hardware implementations", *IEEE Trans. Comp.*, **42**, pp. 281-290, 1993.

[45] B.G. Horne & D.R. Hush, "On the node complexity of neural networks", *Neural Networks*, **7**, pp. 1413-1426, 1994.

[46] K. Hornik, "Approximation capabilities of multilayer feedforward networks", *Neural Networks*, **4**, pp. 251-257, 1991.

[47] K. Hornik, "Some new results on neural network approximation", *Neural Networks*, **6**, pp. 1069-1072, 1993.

[48] K. Hornik, M. Stinchcombe & H. White, "Multilayer feedforward neural networks are universal approximators", *Neural Networks*, **2**, pp. 359-366, 1989.

[49] K. Hornik, M. Stinchcombe & H. White, "Universal approximation of an unknown mapping and its derivatives using multilayer feedforward networks", *Neural Networks*, **3**, pp. 551-560, 1990.

[50] S. Hu, *Threshold Logic*, Berkeley, CA: Univ. California Press, 1965.

[51] S.-C. Huang & Y.-F. Huang, "Bounds on the number of hidden neurons of multilayer perceptrons in classification and recognition", *IEEE Trans. Neural Networks*, **2**, pp. 47-55, 1991.

[52] B. Irie & S. Miyake, "Capabilities of three-layered perceptrons", in *Proc. IEEE Intl. Conf. on Neural Networks ICNN'88*, New York, NY:

IEEE CS Press, 1988, vol. **1**, pp. 641-648.

[53] Y. Ito, "Approximation of functions on a compact set by finite sums of sigmoid functions without scaling", *Neural Networks*, **4**, pp. 817-826, 1991.

[54] Y. Ito, "Approximation capabilities of layered neural networks with sigmoid units on two layers", *Neural Computation*, **6**, pp. 1233-1243, 1994.

[55] L.K. Jones, "A simple lemma on greedy approximation in Hilbert space and convergence rates for projection pursuit regression and neural network training", *Ann. Stat.*, **20**, pp. 608-613, 1992.

[56] H. Katsuura & D.A. Sprecher, "Computational aspects of Kolmogorov's superposition theorem", *Neural Networks*, **7**, pp. 455-461, 1994.

[57] A.H. Khan & E.L. Hines, "Integer-weight neural networks", *Electr. Lett.*, **30**, pp. 1237-1238, 1994.

[58] P. Koiran, "On the complexity of approximating mappings using feedforward networks", *Neural Networks*, **6**, pp. 649-653, 1993.

[59] A.N. Kolmogorov, "On the representation of continuous functions of many variables by superposition of continuous functions of one variable and addition", *Dokl. Akad. Nauk SSSR*, **114**, pp. 953-956, 1957. English transl., *Trans. American Math. Soc.*, **2**, pp. 55-59, 1963.

[60] A.H. Kramer, "Array-based analog computation; principles, advantages and limitations", in *Proc. Microelectronics for Neural Networks MicroNeuro'96*, Los Alamitos, CA: IEEE CS Press, 1996, pp. 68-79.

[61] V. Kůrková, "Kolmogorov's theorem and multilayer neural networks", *Neural Networks*, **5**, pp. 501-506, 1992.

[62] V. Kůrková, P.C. Kainen & V. Kreinovich, "Estimates of the number of hidden units and variations with respect to half-spaces", *Neural Networks*, **10**, pp. 1061-1068, 1997.

[63] Y. LeCun, "Models connexionistes de l'apprentisage", *MSc thesis*, Univ. Pierre et Marie Curie, Paris, France, 1987.

[64] M. Leshno, V.Y. Lin, A. Pinkus & S. Schocken, "Multilayer feedforward neural networks with a nonpolynomial activation function can approximate any function", *Neural Networks*, **6**, pp. 861-867, 1993.

[65] R.P. Lippmann, "An introduction to computing with neural nets", *IEEE ASSP Mag.*, **4**, pp. 4-22, 1987.

[66] O.B. Lupanov, "The synthesis of circuits from threshold elements", *Problemy Kibernetiki*, **20**, pp. 109-140, 1973.

[67] R.D. Mason & W. Robertson, "Mapping Hierarchical Neural Networks to VLSI Hardware," *Neural Networks*, **8**, pp. 905-913, 1995.

[68] H.N. Mhaskar & C. Micchelli, "Approximation by superposition of sigmoidal and radial basis functions", *Advs. in Appl. Maths.*, **13**, pp. 350-373, 1992.

[69] H.N. Mhaskar & C. Micchelli, "Dimension independent bounds on the degree of approximation by neural networks", *IBM J. Res. & Dev.*, **38**, pp. 277-283, 1994.

[70] G.A. Miller, "The magical number seven, plus or minus two: some limits on our capacity for processing information", *Psych. Rev.*, **63**, pp. 71-97, 1956.

[71] R.C. Minnik, "Linear-input logic", *IRE Trans. Electr. Comp.*, **10**, pp. 6-16, 1961.

[72] J. Myhill & W.H. Kautz, "On the size of weights required for linear-input switching functions", *IRE Trans. Electr. Comp.*, **10**, pp. 288-290, 1961.

[73] E.I. Neciporuk, "The synthesis of networks from threshold elements", *Soviet Mathematics—Doklady*, **5**, pp. 163-166, 1964. English transl., *Automation Express*, **7**, pp. 27-32 & **7**, pp. 35-39, 1964.

[74] M. Nees, "Approximate versions of Kolmogorov's superposition theorem, proved constructively", *J. Comp. & Appl. Math.*, **54**, pp. 239-250, 1994.

[75] M. Nees, "Chebyshev approximation by discrete superposition: application to neural networks", *Advs. in Comp. Maths.*, **5**, pp. 137-152, 1996.

[76] I. Parberry, *Circuit Complexity and Neural Networks*. Cambridge: MIT Press, 1994.

[77] J. Park & I.W. Sandberg, "Universal approximation using radial-basis-function networks", *Neural Computation*, **3**, pp. 246-257, 1991.

[78] J. Park & I.W. Sandberg, Approximation and radial-basis-function networks, *Neural Computation*, **5**, pp. 305-316, 1993.

[79] H. Paugam-Moisy, "Optimisation des réseaux des neurones artificiels", *PhD Dissertation*, LIP (http://www.ens-lyon.fr/LIP/publis.us.html), École Normale Supérieure de Lyon, Lyon, France, 1992.

[80] D.S. Phatak & I. Koren, "Connectivity and performances tradeoffs in the cascade correlation learning architecture", *IEEE Trans. Neural Networks*, **5**, pp. 930-935, 1994.

[81] T. Poggio & F. Girosi, "A theory of networks for approximation and learning", *Tech. Rep. AI Memo 1140* (ftp://publications.ai.mit.edu/ai-publications/1000-1499/AIM-1140.ps.Z), MIT, 1989. Short version: Networks for approximation and learning, *Proc. IEEE*, **78**, pp. 1481-1497, 1990.

[82] P. Raghavan, "Learning in threshold networks: a computational model and applications", *Tech. Rep. RC 13859*, IBM Res., 1988. Also in *Proc. Intl. Conf. on Comp. Learning Theory COLT'88*, New York, NY: ACM Press, 1988, pp. 19-27.

[83] N.P. Red'kin, "Synthesis of threshold circuits for certain classes of Boolean functions", *Kibernetika*, **5**, pp. 6-9, 1970. English transl., *Cybernetics*, **6**, pp. 540-544, 1973.

[84] V.P. Roychowdhury, A. Orlitsky & K.-Y. Siu, "Lower bounds on threshold and related circuits via communication complexity", *IEEE Trans. Info. Theory*, **40**, pp. 467-474, 1994.

[85] F. Scarselli & A.C. Tsoi, "Universal approximation using feedforward neural networks: a survey of some existing methods, and some new results", *Neural Networks*, **11**, pp. 15-37, 1998.

[86] C. Shannon, "The synthesis of two-terminal switching circuits", *Bell Sys. Tech. J.*, **28**, pp. 59-98, 1949.

[87] K.-Y. Siu, V.P. Roychowdhury & T. Kailath, "Depth-size tradeoffs for neural computations", *IEEE Trans. Comp.*, **40**, pp. 1402-1412, 1991.

[88] E.D. Sontag, "Shattering all sets of k points in 'general position' requires $(k-1)/2$ parameters", *Report SYCON 96-01*, Maths. Dept., Rutgers Univ., 1996. Also in *Neural Computation*, **9**, pp. 337-348, 1997.

[89] D.A. Sprecher, "On the structure of continuous functions of several variables", *Trans. American Math. Soc.*, **115**, pp. 340-355, 1965.

[90] D.A. Sprecher, "On the structure of representations of continuous functions of several variables as finite sums of continuous functions of one variable", *Proc. American Math. Soc.*, **17**, pp. 98-105, 1966.

[91] D.A. Sprecher, "A universal mapping for Kolmogorov's superposition theorem", *Neural Networks*, **6**, pp. 1089-1094, 1993.

[92] D.A. Sprecher, "A numerical implementation of Kolmogorov's superpositions", *Neural Networks*, **9**, pp. 765-772, 1996.

[93] D.A. Sprecher, "A numerical construction of a universal function for Kolmogorov's superpositions", *Neural Network World*, **6**, pp. 711-718, 1996.

[94] D.A. Sprecher, "A numerical implementation of Kolmogorov's superpositions II", *Neural Networks*, **10**, pp. 447-457, 1997.

[95] M. Stevenson & S. Huq, "On the capability of threshold adalines with limited-precision weights", *Neural Computation*, **8**, pp. 1603-1610, 1996.

[96] J.D. Ullman, *Computational Aspects of VLSI*, Rockville, MA: Comp. Sci. Press, 1984.

[97] S. Vassiliadis, S. Cotofana & K. Berteles, "2-1 Addition and Related Arithmetic Operations with Threshold Logic", *IEEE Trans. Comp.*, **45**, pp. 1062-1068, 1996.

[98] M.R. Walker, S. Haghighi, A. Afghan & L.A. Akers, "Training a limited-interconnect, synthetic neural IC", in D.S. Touretzky (ed.), *Advances in Neural Information Processing Systems*, San Mateo, CA: Morgan Kaufmann, 1989, pp. 777-784.

[99] R.C. Williamson, "ε-entropy and the complexity of feedforward neural networks", in R.P. Lippmann, J.E. Moody & D.S. Touretzky (eds.), *Advances in Neural Information Processing Systems*, San Mateo, CA: Morgan Kaufmann, 1990, pp. 946-952.

[100] J. Wray & G.G.R. Green, "Neural networks, approximation theory, and finite precision computation", *Neural Networks*, **8**, pp. 31-37, 1995.

[101] B.-T. Zhang & H. Mühlenbein, "Genetic programming of minimal neural networks using Occam's razor", *Tech. Rep. GMD 734*, Schloß Birlinghoven, St. Augustin, Germany, 1993. Also in *Complex Systems*, **7**, pp. 199-220, 1993.