

Defect-tolerant nanoelectronic pattern classifiers

Jung Hoon Lee and Konstantin K. Likharev*,†

Stony Brook University, Stony Brook, NY 11794-3800, U.S.A.

SUMMARY

Mixed-signal neuromorphic networks ('CrossNets'), based on hybrid CMOS/nanodevice circuits, may provide unprecedented performance for important pattern classification tasks. The synaptic weights necessary for such tasks may be imported from an external 'precursor' network with either continuous or discrete synaptic weights (in the former case, with the quantization—'clipping'—due to the binary character of the elementary synaptic nanodevices—latching switches.) Alternatively, the weights may be adjusted '*in situ*' (inside the CrossNet) using a pseudo-stochastic method, or set-up using a mixed-mode method partly employing external circuitry. Our calculations have shown that CrossNet pattern classifiers, using any of these synaptic weight adjustment methods, may be remarkably resilient. For example, in a CrossNet with synapses in the form of two small square arrays with 4×4 nanodevices each, the resulting weight discreteness may have a virtually negligible effect on the classification fidelity, while the fraction of defective devices which affects the performance substantially ranges from $\sim 20\%$ to as high as 90% (!), depending on the training method. Copyright © 2007 John Wiley & Sons, Ltd.

Received 15 February 2007

KEY WORDS: nanoelectronics; nanowires; CMOS; CMOL; hybrid circuits; neuromorphic networks; pattern classification; training; learning; defect tolerance

1. INTRODUCTION

It is generally accepted now [1] that the current VLSI digital circuit paradigm, based on a combination of lithographic patterning, CMOS technology, and Boolean logic, cannot be extended into the few-nm minimum size range without introducing new concepts and/or devices. On the other

*Correspondence to: Konstantin K. Likharev, Stony Brook University, Stony Brook, NY 11794-3800, U.S.A.

†E-mail: klicharev@notes.cc.sunysb.edu

Contract/grant sponsor: U.S. Air Force Office of Scientific Research

Contract/grant sponsor: U.S. Microelectronics Advanced Research Corporation *via* the FENA Center

Contract/grant sponsor: Division of Computer-Communication Research, U.S. National Science Foundation

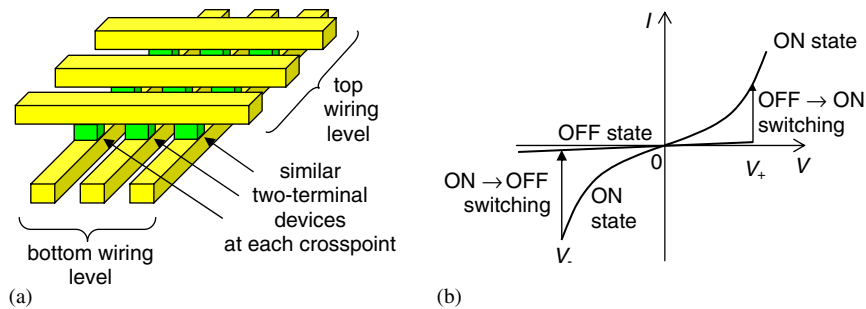


Figure 1. (a) Crossbar structure and (b) I - V curve of a two-terminal latching switch (schematically).

hand, the experimentally demonstrated or rationally envisioned sub-10-nm devices (for a review, see, e.g. Reference [2]) fall into two distinct categories:

- (i) multi-terminal devices with functionality comparable with that of a silicon MOSFET (in particular, with substantial voltage gain) and
- (ii) simple, two-terminal devices.

So far, only the devices of the second group could be fabricated with the necessary (sub-nm) accuracy using potentially inexpensive techniques, because they may have just one critical dimension (distance between the electrodes) which may be readily controlled by, e.g. film deposition or oxidation rate. Unfortunately, the functionality of these diode-like devices is hardly sufficient for the implementation of valuable integrated circuits.

Hybrid CMOS/nanodevice circuits offer an effective resolution of this situation. In such circuits (see, e.g. recent reviews [3, 4]), a layer of two-terminal nanodevices is used as an add-on to a CMOS subsystem. Such a combination allows both the functionality, flexibility, and reliability of CMOS circuits (and all the enormous infrastructure accumulated by the electronic industry for their design and fabrication) and the unparalleled potential density of nanodevices (with their miniscule footprint) to be used to the full extent.

Virtually all realistic proposals of such hybrid CMOS/nanodevice circuits are based on the crossbar structures (Figure 1(a)) with latching switches (a.k.a. 'programmable diodes') formed at each crosspoint. The functionality of such two-terminal devices is illustrated in Figure 1(b). At low applied voltages, the devices behave as usual resistors or diodes, but higher voltages may be used to switch them between low-resistive (ON) and high-resistive (OFF) states. Numerous devices with such functionality have been already demonstrated using several materials, notably including amorphous metal-oxide films (see, e.g. References [5, 6]), relatively thick organic films (both with [7, 8] and without [9, 10] embedded metallic clusters), self-assembled molecular monolayers [11–13], and thin chalcogenide layers [14, 15]. The physics of the ON–OFF switching in these devices is still a matter of substantial discussion, with the reversible filament formation most probable for organic systems, and trapped electric charge accumulation looking like the most plausible candidate for amorphous oxide films. In both interpretations, the conductance is due to some random active conducting centres (filaments or hopping percolation paths) separated by distances of the order of a few nanometers. In order to be reproducible, the device should have a large number of such centres. This is why the extension of the excellent reproducibility demonstrated for such statistical crosspoint devices with the lateral size $F > 100$ nm [5] to the most

interesting range $F < 10$ nm presents a challenge. This problem may be addressed using uniform self-assembled monolayers of specially designed molecules [16] implementing single-electron latching switches [17].[‡] A major challenge here is the reproducibility of the interface between the monolayer and the second (top) metallic electrode, because of the trend of the metallic atoms to diffuse inside organic monolayers during the electrode deposition [22]. Recent very encouraging progress towards solving this problem has been obtained using an intermediate layer of a conducting polymer [46].

The crossbar geometry (Figure 1(a)) allows the state of each latching switch to be set up and read out individually.[§] The main technological advantage of this structure is that the crossbar may be formed by advanced patterning methods (e.g. nanoimprint [22, 23]) which may be potentially extended to a few-nm half-pitch F_{nano} ,[¶] bringing the latching switch density beyond 10^{12} devices per cm^{-2} . At this approach, interfacing the nanowire crossbar with the CMOS subsystem (which has to perform demultiplexing, readout signal sensing, error correction and other peripheral functions), becomes a challenge, because the advanced patterning technologies lack nanoscale alignment (overlay) of sequential layers. Several initial suggestions for forming such interfaces at the crossbar periphery [4] do not seem very practicable—see Reference [23] for their critical review. Recently, our group suggested [2, 23] a new approach (dubbed ‘CMOL’) in which the interface is provided all over the chip surface, using pins with the CMOS pitch but nanoscale-sharp tips (Figure 2). The main feature of the CMOL topology is that it provides a unique access from the CMOS subsystem to each crossbar nanowire (whether in the bottom or the top layer) with the theoretical 100% fabrication yield even in the absence of *any* alignment between the CMOS and crossbar subsystems.^{||}

Calculations have shown that CMOL technology may enable terabit-scale ‘resistive’ (or ‘crossbar’) memories with sub-100-ns access time and defect tolerance above 10% [24] and FPGA-like reconfigurable logic circuits with density at least two orders of magnitude higher than that of their CMOS counterparts fabricated with same design rules, at even higher defect tolerance (above 20%) and manageable power dissipation [25]. However in the long run, the most important application of this technology may be in advanced information processing with mixed-signal neuromorphic networks—‘CrossNets’ [16, 17, 23, 26].

Figure 3 shows the generic (feedforward, binary-weight) CMOL CrossNet structure. CMOS-implemented somatic cells apply their analog output voltages V_j to ‘axonic’ nanowires. If the latching switch at the crosspoint of an axonic wire with a perpendicular ‘dendritic’ wire leading to cell i is in its ON state with relatively high conductance G , it supplies current $I_{ij} \approx GV_j$ into the latter wire, changing the electric charge Q_i of the wire and hence the input signal of the post-synaptic somatic cell. Since each dendritic wire contacts several (M) synapses (Figure 3),

[‡]Metal-based, low-temperature prototypes of such switches, with multi-hour state retention times, have been demonstrated experimentally [2, 18, 19]. However, so far molecular implementations have been only demonstrated (see, e.g. References [20, 21]) for the main component of the device, the single-electron transistor [2], rather than for the latching switch as the whole.

[§]For example, in order to turn the switch ON, the two wires leading to the device are fed by voltages $\pm V_{\text{WRITE}}$, with $V_{\text{WRITE}} < V_+ < 2V_{\text{WRITE}}$. (The right inequality ensures that this operation does not disturb the state of ‘semi-selected’ switches contacting just one of the biased nanowires.) The reciprocal ON \rightarrow OFF switching is performed similarly using threshold V_- (Figure 1(b)).

[¶]Prototype crossbars with $F_{\text{nano}} = 30$ nm have already been demonstrated [13].

^{||}Note that the last statement is only true for the recently introduced [25] version of the CMOL interface in which each ‘blue’ pin, reaching to the upper nanowire level, intentionally interrupts a bottom level wire—see Figure 2(a).

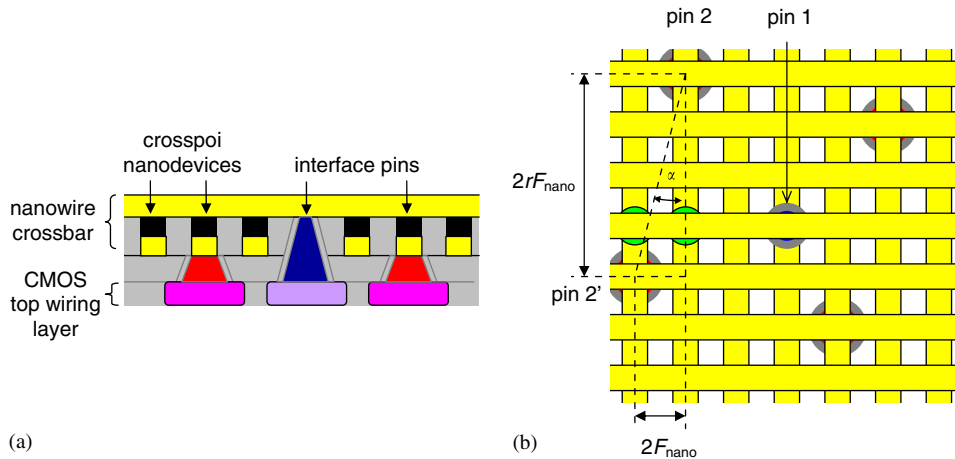


Figure 2. Structure of a CMOL circuit: (a) schematic side view and (b) top view. For clarity, the last panel shows only two adjacent crosspoint devices which may be addressed *via* interface pin pairs $\{1, 2\}$ and $\{1, 2'\}$. The figure shows that due to the specific rotation angle $\alpha = \arctan(1/r)$, where r is an integer, between the nanowire crossbar and the square meshes of interface pins, each nanodevice may be accessed individually from the CMOS subsystem.

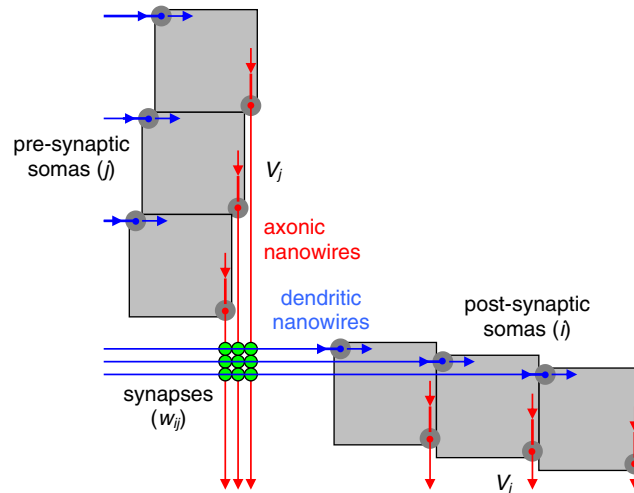


Figure 3. Structure of a generic (feedforward, binary-synapse) CrossNet. Physically, the axonic nanowires (shown red), and dendritic nanowires (shown blue) are similar. For clarity, only a fraction of elementary synapses (latching switches) are shown (by green circles). Actually, the devices form a continuous 2D array covering all the chip area including the area of underlying, CMOS-implemented somatic cells (shown grey). Note the open-circuit terminations of axonic and dendritic lines. Due to these terminations the somas do not communicate directly (but only *via* synapses); the terminations also limit the nanowire segment length and hence the cell connectivity M .

the current addition provides a natural passive analog summation of signals from M pre-synaptic cells.

The most remarkable feature of the CMOL CrossNet structure is that connectivity M in these quasi-2D circuits may be large, which seems necessary for all but most rudimentary applications of neural networks—see, e.g. References [27–29]. In order to increase M with a given nanowire crossbar, it is sufficient to reduce somatic cell *density*. In contrast, the cell *distribution*, at fixed density, defines the network configuration, i.e. their connectivity graph. For example, it is straightforward to design [26] both layered networks ('FlossBars') which differ from the usual multilayer perceptrons [27, 28] only by a finite connectivity, and 'interleaved' structures ('InBars') which are most natural for the implementation of recurrent networks.

The largest motivation for the CMOL CrossNet development is the enormous estimated performance of these networks. Apparently, this is the first hardware which eventually may overcome human cerebral cortex in cell density per unit area (at the similar average connectivity $M \approx 10^4$ [30]), beating the biological prototype in speed by at least four orders of magnitude at a readily manageable power dissipation of the order of 1 W/cm². The comparison between CrossNets and the usual artificial neural networks (implemented as software codes run on usual digital computers) is also very impressive. For example, we have shown [31] that a CMOL CrossNet chip of a modest (sub-1-cm²) size, dissipating ~ 10 W, can use the well-developed classification algorithms run independently in 4000 panels (of 32×32 pixels each) to provide a high-fidelity recognition of a human face in a large crowd in just 100 μ s (compatible with online operation). This number is to be compared with a few-hour time necessary if the same task is being performed by a modern CMOS-based microprocessor.

However, in order to use this potential performance edge, several challenges have to be met. Even leaving alone the major technological task of nanodevice and nanowire scaling into the sub-10-nm range, the specific properties of CMOL hardware create additional problems with CrossNet training in comparison with the software-implemented neural networks. Indeed, the elementary synapse (nanoscale latching switch):

- (i) can only provide binary synaptic weight;
- (ii) may have a substantial probability q of being defective; and
- (iii) may be adjusted only *via* the two nanowires it has been formed between.

For some tasks, these challenges may be readily met. For example, in Little–Hopfield-type symmetric recurrent networks, using binary synaptic weights leads to a very limited ($\sim 30\%$) reduction of the network capacity [28], while the defect tolerance may be very high (above 80%, at the 50% capacity reduction), and since the weight calculation is straightforward, they may be readily imported from an external computer [26].

However, other important applications, in particular feedforward perceptrons used for pattern classification [29], are more demanding and require, in particular, multi-valued synaptic weights. The main goal of this paper is to show that weight clipping and modest amount of defects do not prevent CMOL CrossNet from being trained and used as defect-tolerant pattern classifiers. In Sections 2 and 3 we will address, respectively, the procedure of synaptic weight import from a precursor network with continuous weights, and the clipping and defect tolerance of CrossNet perceptrons with imported weights. The second option is to use precursor networks with discrete weights, avoiding the clipping at import. We will discuss this approach, which gives slightly inferior results for perfect CrossNets but much better defect tolerance, in Section 4. Section 5 is devoted to a discussion of '*in situ*' training of CrossNet classifiers; this method has showed very

high defect tolerance on small problems, but does not seem scalable to tasks with larger data sets. The last fact has motivated us to work on the mixed-mode training (Section 6) which provides both scalability and high defect tolerance, using smaller external tutors. Finally, in Section 7 we summarize our results and discuss the most urgent unsolved problems of CrossNet development.

2. SYNAPTIC WEIGHT IMPORT

The most natural way to produce an L -level synaptic weight in CMOL CrossNets is to use a square array of $n \times n$ similar latching switches (Figure 4(a)). In the feedforward signal propagation mode the output currents of all the devices are summed up, so that the total output signal is proportional to the number N of switches in the ON state ($0 \leq N \leq n^2$), giving $n^2 + 1$ possible values of w_{ij} . For the CrossNet geometry, it is more natural to use two such arrays, one giving a positive and another negative contribution to the dendritic signal, thus yielding $L = 2n^2 + 1$ quantized weight levels [26].

The weights may be imported from outside, for example, from a homomorphic ‘precursor’ network with quasi-continuous synaptic weights. At the import process [26], each weight is naturally rounded (‘clipped’) to one of the L quantized values corresponding to the possible state of the CrossNet synapse, giving clipped values $w_N = w_{\max}(N/n^2)$, where $N = -n^2, -n^2 + 1, \dots, +n^2$ is the effective number of binary switches in their ON state.**

Generally, the import may be accomplished by setting each latching switch individually—see Footnote §—because CMOL topology allows the CMOS subsystem to address each nanowire and hence each nanodevice. However, in order to simplify the subsystem, the whole ‘composite synapse’

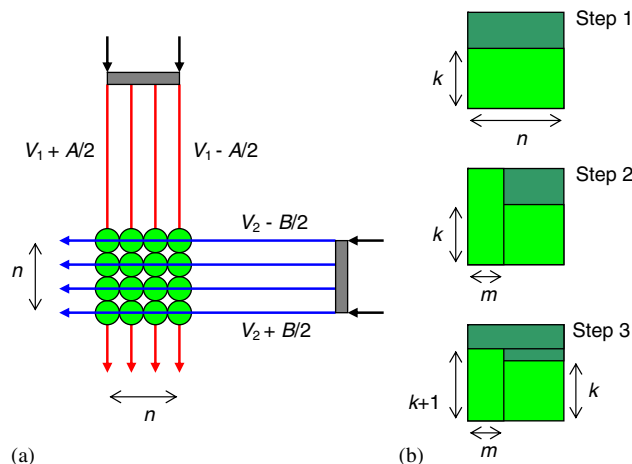


Figure 4. (a) A multi-valued synapse and (b) a 3-time-step weight import into it. Dark-green and light-green colours show the areas occupied by latching switches turned, respectively, OFF and ON.

**The synaptic weight amplitude w_{\max} may be regulated by physical parameters of the network, including conductance G of the latching switch in its ON state.

(the $n \times n$ array) may be adjusted simultaneously by creating linear gradients of voltages applied to each nanowire (Figure 4). If the gradients are equal ($A = B$) the OFF \rightarrow ON switching condition $V = V_+$ creates a boundary, with a 45° slope, which can be moved through the array, changing N —see Figure 10 of Reference [26] and its discussion. However, at this import technique the switching boundary may reach several (up to n) devices simultaneously, thus effectively reducing the number of controllable levels.

Figure 4(b) shows a way around this problem. In this approach, the weights are imported in three sequential time steps (after the '0th' step of setting all the switches to OFF state). At step 1, the voltage gradient is created only along one group (say, dendritic) nanowires: $A = 0, B > 0$. The resulting vertical gradient of applied voltage creates a horizontal OFF \rightarrow ON switching line which turns on $n \times k$ lower switches, where k is the first integer in the module- n expression of the desired number of ON devices: $N = kn + m$. At step 2, $A > 0, B = 0$, and the resulting horizontal gradient of applied voltage is used to set ON all devices in the $n \times m$ left rectangle. (Of those switches, $m \times k$ have already been in the ON state after step 1.) Finally, at step 3, $A = 0, B > 0$ again, and the vertical gradient is used to drive the ON \rightarrow OFF switching boundary (using the threshold V_- , see Figure 1(b)) one line above that used at step 1. This turns off all the switches but m in the vertical strip, and brings the total number of ON devices to the desired number $N = kn + m$.

3. CONTINUOUS WEIGHT IMPORT: CLIPPING AND DEFECT TOLERANCE

A useful initial insight on the impact of weight import on the network fidelity may be obtained for the analytically tractable case of a simple perceptron [27–29] with M inputs, no hidden layers and one or several output signals

$$O_i = g(y_i), \quad y_i = \sum_{j=1}^M w_{ij}x_j \quad (1)$$

The class of a certain input vector $\{x_j\}$ is usually decided by the comparison of output signals O_i . For a monotonic activation function $g(y)$, this is equivalent to the comparison of sums y_i . Let us first treat the network as a statistical ensemble, with both signal components x_j and synaptic weights w_{ij} approximately treated as independent random variables. If, for the notation simplicity, we consider sign-symmetric inputs x_j and weights w_{ij} , then

$$\langle x_j \rangle = \langle w_{ij} \rangle = \langle x_j x_{j'} \rangle = \langle w_{ij} w_{i'j'} \rangle = \langle w_{ij} x_{j'} \rangle = 0 \quad (2)$$

As a result, the spread of the output signals (in the case of perfect weight values) may be calculated as

$$\langle y^2 \rangle = M \langle x^2 \rangle \langle w^2 \rangle \quad (3)$$

Even at perfect nanodevices, the clipping procedure at weight import changes the weights in comparison with their initial values w (Figure 5(a)):

$$w_c = \begin{cases} -w_{\max} & \text{for } w < -w_{\max} \\ w_N & \text{for } |w - w_N| < w_{\max}/2n^2 \\ w_{\max} & \text{for } w > w_{\max} \end{cases} \quad (4)$$

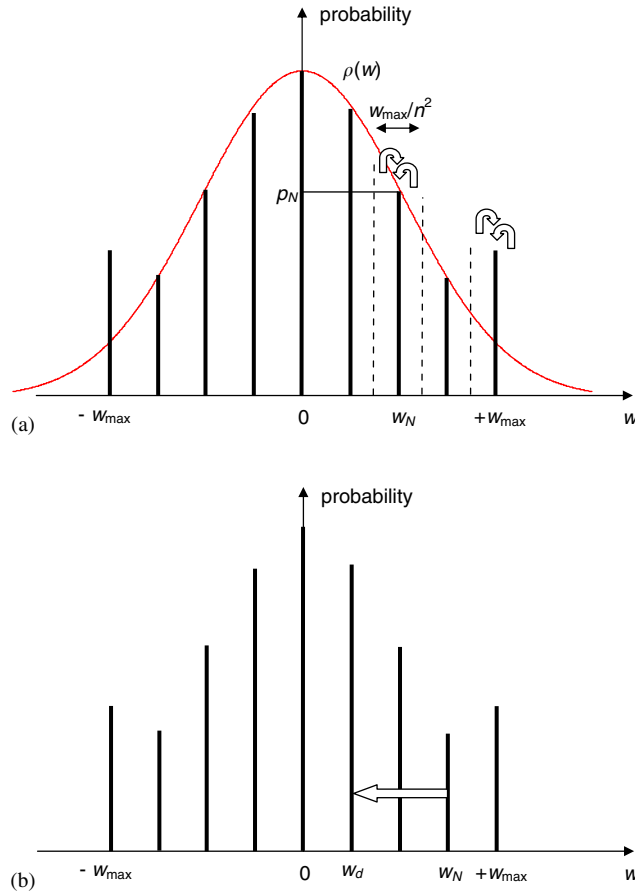


Figure 5. Change of weights due to: (a) clipping and (b) stuck-on-close-type defects—schematically.

which creates the output signal variance^{††}

$$\langle \delta y_c^2 \rangle = M \langle x^2 \rangle \langle \delta w_c^2 \rangle, \quad \delta w_c \equiv w_c - w \tag{5}$$

The relative weight perturbation due to clipping may be characterized by the ratio

$$R_c^2 \equiv \frac{\langle \delta w_c^2 \rangle}{\langle w_c^2 \rangle} = \frac{\int_{-\infty}^{+\infty} \delta w_c^2 \rho(w) \, dw}{\int_{-\infty}^{+\infty} w_c^2 \rho(w) \, dw} \tag{6}$$

where $\rho(w)$ is the probability distribution of the ‘perfect’ (unclipped) synaptic weights. Numerical simulations of perceptrons trained to perform real classification tasks show that this distribution

^{††}Strictly speaking, the last average in Equation (5), as well as all the averages denoted $\langle \dots \rangle$ below, is carried out over a sub-ensemble of random weight sets, which nominally give a certain output vector y_i .

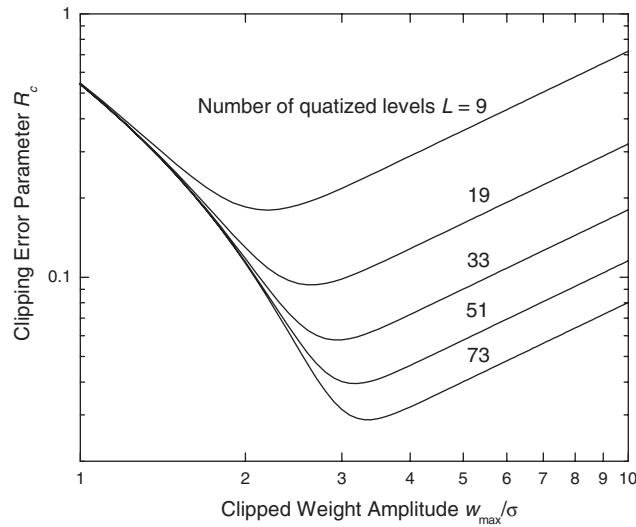


Figure 6. Parameter R_c of simple perceptron weight perturbation due to clipping, as a function of the clipped weight amplitude width w_{max} (normalized to the width σ of the initial continuous-weight distribution) for several values of the number $L = 2n^2 + 1$ of quantized levels.

is typically close to the Gaussian. (At least this is a much better approximation than the uniform distribution used in the earlier analyses [26, 32, 33] of the weight clipping effects.) In this case, in the most important limit $L \gg 1$, Equations (3) and (5) yield

$$R_c^2 = \frac{\frac{\mu^2}{12n^4} \int_0^\mu \exp\left(-\frac{\xi^2}{2}\right) d\xi + \int_\mu^\infty (\xi - \mu)^2 \exp\left(-\frac{\xi^2}{2}\right) d\xi}{\int_0^\mu \xi^2 \exp\left(-\frac{\xi^2}{2}\right) d\xi + \mu^2 \int_\mu^\infty \exp\left(-\frac{\xi^2}{2}\right) d\xi} \tag{7}$$

where $\mu \equiv w_{max}/\sigma$, and σ^2 is the initial distribution’s variance. (All the integrals can be readily expressed *via* the error function.) The error parameter R_c as a function of μ has a minimum (Figure 6) corresponding to the optimal choice of the quantized weight distribution width.^{‡‡}

Let us use the same approach for the analysis of the effect of fabrication-induced defects. We will use the same defect model which has been accepted in other studies of hybrid semiconductor/nanodevice circuits [24, 25, 34, 35], namely that the crosspoint latching switch defects are independent (with probability q) and are equivalent to ‘stuck-at-open’ faults, i.e. that the bad device is either missing (leaving the nanowires disconnected) or is always in the OFF state (Figure 1).

^{‡‡}The reason for that is as follows: if the clipped weight amplitude w_{max} (and hence parameter μ) is small, the error is large due to clipping of the weight distribution tails. On the other hand, if the amplitude is made so large that it covers virtually all the essential distribution of the initial weights, the weight rounding of internal values becomes the most essential source of the total error now which grows with w_{max} .

At the current stage of latching switch development, such defects dominate, though an analysis of the opposite, ‘stuck-at-close’ defects is an important goal for future research.

For our ‘composite’ synapse of $2n^2 = L - 1$ elementary latching switches, the defects may result in the reduction of the number of ON synapses (Figure 5(b)), from the number N corresponding to the nominal value $w_N = w_{\max}N/n^2$ of the clipped weight, to some random number N_d with the binomial probability distribution

$$\langle N_d \rangle_d = N(1 - q), \quad \langle N_d^2 \rangle_d = N^2(1 - q)^2 + |N|q(1 - q) \tag{8}$$

where $\langle \dots \rangle_d$ means averaging over the ensemble of random defects (at fixed N). In order to calculate statistical properties of the full weight perturbation

$$\delta w \equiv w_d - w = (w_d - w_N) + (w_N - w) = \delta w_d + \delta w_c = (w_{\max}/n^2)(N_d - N) + \delta w_c \tag{9}$$

we need to average them also over the clipped weight ensemble

$$\langle \langle \dots \rangle_d \rangle = \sum_{N=-n^2}^{+n^2} p_N \langle \dots \rangle_d \tag{10}$$

where p_N is the probability distribution of N due to clipping (Figure 5(a)). This distribution may be obtained by the integration of the pre-clipping distribution function over the range of w clipped to the corresponding value of N —see Equation (4). For $L \gg 1$, we get

$$p_N = \frac{1}{\sqrt{2\pi}} \begin{cases} \frac{\mu}{n^2} \exp\left(-\frac{\mu^2 N^2}{2n^4}\right), & |N| < n^2 \\ \int_{\mu}^{\infty} \exp\left(-\frac{\xi^2}{2}\right) d\xi, & |N| = n^2 \end{cases} \tag{11}$$

Using these formulas, we may calculate the variance of the total perturbation

$$\langle \langle \delta w^2 \rangle_d \rangle = \langle \langle \delta w_d^2 \rangle_d \rangle + \langle \delta w_c^2 \rangle + 2\langle \langle \delta w_c \delta w_d \rangle_d \rangle = \langle \langle \delta w_d^2 \rangle_d \rangle + \langle \delta w_c^2 \rangle + 2q\langle \langle w_c \delta w_d \rangle_d \rangle \tag{12}$$

taking into account that at $L \gg 1$ the summation over the ‘internal’ values of N (with $N^2 < n^4$) may be replaced by integration. The result is $R^2 = \langle \langle \delta w^2 \rangle_d \rangle / \langle \langle w_d^2 \rangle_d \rangle$, with

$$\langle \langle \delta w^2 \rangle_d \rangle = \sigma^2 \sqrt{\frac{2}{\pi}} \left\{ \begin{aligned} & \left[\frac{1}{12} \left(\frac{\mu}{n^2}\right)^2 \int_0^{\mu} \exp\left(-\frac{\xi^2}{2}\right) d\xi + \int_{\mu}^{\infty} (\xi - \mu)^2 \exp\left(-\frac{\xi^2}{2}\right) d\xi \right] \\ & + q^2 \left[\int_0^{\mu} \xi^2 \exp\left(-\frac{\xi^2}{2}\right) d\xi + \mu^2 \int_{\mu}^{\infty} \exp\left(-\frac{\xi^2}{2}\right) d\xi \right] \\ & + q(1 - q) \frac{\mu}{n^2} \left[\int_0^{\mu} \xi \exp\left(-\frac{\xi^2}{2}\right) d\xi + \frac{\mu}{\sigma} \int_{\mu}^{\infty} \exp\left(-\frac{\xi^2}{2}\right) d\xi \right] \\ & + 2q\mu \int_{\mu}^{\infty} (\xi - \mu) \exp\left(-\frac{\xi^2}{2}\right) d\xi \end{aligned} \right\} \tag{13}$$

$$\langle \langle w_d^2 \rangle \rangle = \sigma^2(1-q)\sqrt{\frac{2}{\pi}} \left\{ \begin{aligned} & (1-q) \int_0^\mu \xi^2 \exp\left(-\frac{\xi^2}{2}\right) d\xi + q \frac{\mu}{n^2} \int_0^\mu \xi \exp\left(-\frac{\xi^2}{2}\right) d\xi \\ & + \left[(1-q)\mu^2 + q \frac{\mu}{n^2} \right] \int_\mu^\infty \exp\left(-\frac{\xi^2}{2}\right) d\xi \end{aligned} \right\} \quad (14)$$

Figure 7 shows the total perceptron perturbation parameter R (due to both clipping and defects), as a function of μ , for several values of the bad device fraction q and two practicable values of L , corresponding to $n = 3$ and 4 . Just as in the case of pure clipping, R can be minimized by an appropriate choice of μ for each L and q . Figure 8 shows this minimum value of R , as well as the necessary value of w_{\max} , as a function of q , for several realistic values of L .

These results seem rather alarming, because the probability for the perturbation to result in the wrong sign of an output signal is given by $\varepsilon = (1/\pi) \arctan R$ (see, e.g. Reference [28]). This means, for example, that if a CrossNet with the realistic value $n = 4$ (i.e. $L = n^2 + 1 = 33$) obeyed the simple theory developed above, the defect fraction $q = 20\%$ would lead to $R \approx 0.3$ and hence to $\sim 10\%$ of outputs with a wrong sign. However, the actual situation may be better. Indeed, the calculations described above have been carried out in the assumption of statistical independence of input signals x_j and synaptic weights w_{jk} . Actually, perceptrons training makes the weights strongly correlated with input signals. For example, in a well-trained ‘Winner Takes All’ network (where a certain class C_p is coded by the dominance of p th output over others, i.e. by the relation $y_p > y_i$ for all $i \neq p$), there is always a substantial gap between y_p and $\max_i(y_i)$ for all input vectors. In this case, a small weight perturbation, which leads to a proportional broadening of the output signal distributions, with $\langle \delta y^2 \rangle / \langle y^2 \rangle \sim \langle \delta w^2 \rangle / \langle w^2 \rangle = R^2$, should not give a substantial error unless $\langle \delta y^2 \rangle$ becomes comparable with the nominal difference $y_p - \max_i[y_i]$. Though we are unaware of any analytical theory of the resulting errors, one can expect that if the output signal broadening is close to Gaussian, a fair estimate of the error is given by the error function

$$\varepsilon \sim \frac{1}{\sqrt{2\pi(2aR)^2}} \int_{-\infty}^0 \exp\left(-\frac{(\Delta + \xi)^2}{2(2aR)^2}\right) d\xi \quad (15)$$

where $\Delta = (y_p - \max[y_{i \neq p}]) / y_p$, and a is a coefficient of the order of 1. According to this formula (with optimistic values $a = 1$, $\Delta = 2$), for the same case $L = 33$, $q = 20\%$ (giving $2R \approx 0.6$), we get a crude estimate $\varepsilon \sim 5\%$. Such an error would be acceptable for nearly all perceptron applications, since the generalization errors of ‘perfect’ (continuous-weight) networks trained for pattern classification problems are typically of the order of a few percent [27–29].

The approximate nature of this estimate has urged us to carry out a more direct study of the clipping and defect impact on pattern classification by perceptrons. In particular, we have carried out a numerical Monte Carlo simulation of a simple but representative case of a perceptron with one hidden layer trained to classify handwritten digits of the MNIST database [36, 37]. This set consists of 60 000 training plus 10 000 test images of 10 digits (0–9). Each image has $28 \times 28 = 784$ pixels (with 256 levels of grey) so that for its classification we have used a 784-784-10 cell perceptron.^{§§} The somatic cells are just non-linear amplifiers with the activation function $x_i = \tanh(gy_i)$. The normalized gain g has been fixed at the level $2(3/M)^{1/2}$ (where $M = 784$

^{§§}The exact number of units in the hidden layer does not affect the performance significantly [36].

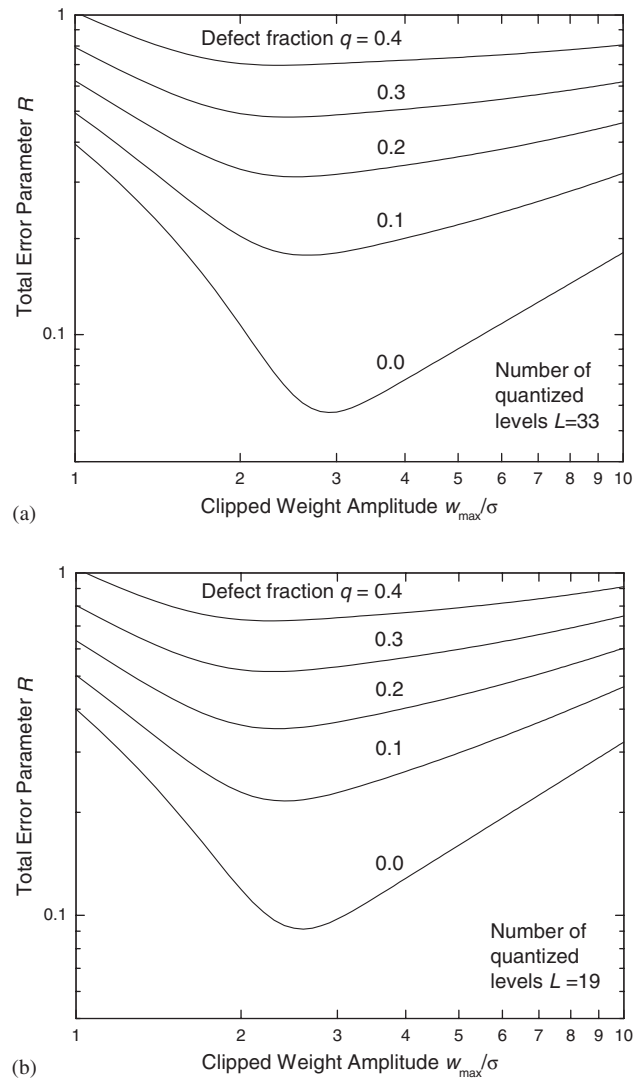


Figure 7. The simple perceptron total perturbation parameter R (due to both clipping and defects) as a function of the clipped weight distribution width w_{\max} , for two typical values of L and several values of the bad device fraction q .

is the number of cells in the previous layer) which gives a reasonable (moderate) degree of the activation function non-linearity. The class of the input vector was determined by the number of the output channel giving the largest signal (the ‘Winner Takes All’ rule). A homomorphic perceptron with continuous weights has been trained by the usual error backpropagation method [27, 28]. The pre-training weights were random, with a uniform distribution over the range $[-1, +1]$, but the

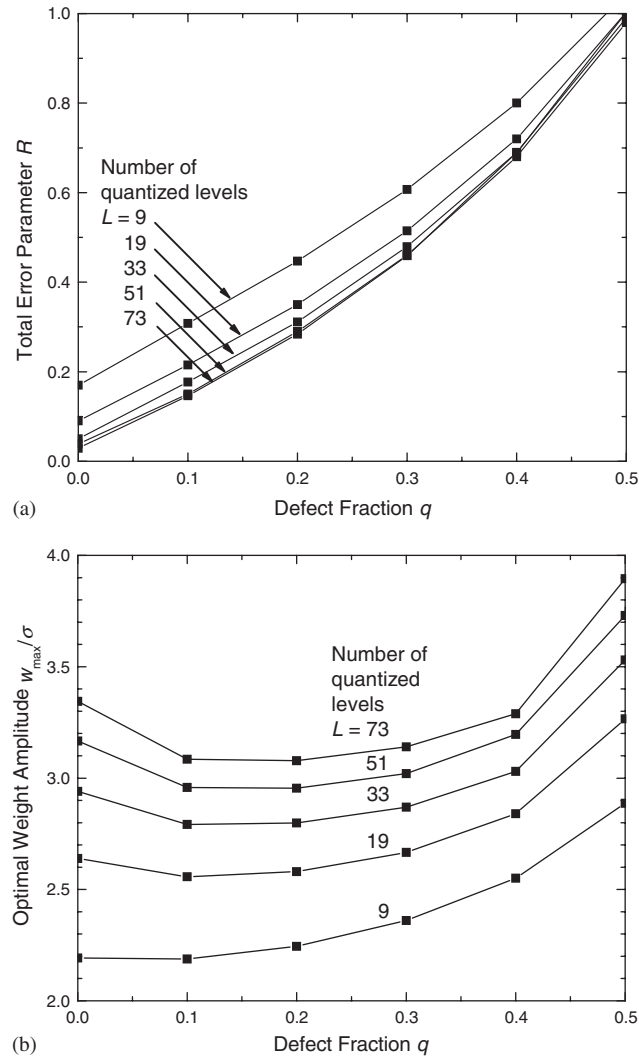


Figure 8. (a) The minimum value of the total perceptron error parameter R and (b) the optimal value of the w_{\max}/σ ratio as functions of q , for several values of L .

final (post-training) distribution was closer to Gaussian, especially for the synapses between the first and hidden layers.

After training, the weights have been imported, with the corresponding clipping to simulated CrossNets with random position of bad latching switches. At this procedure, for a given number of quantization levels L , we could select an arbitrary clipped weight amplitude w_{\max} . Moreover, since precursor network properties do not depend on the scale of weights of synapses between the hidden and output layers, we could multiply these weights, before clipping, by an arbitrary factor c . (The effect of this parameter is similar to that of w_{\max} , but in CMOL hardware it is much more

natural to keep the latter parameter global.) For a substantial defect concentration q we have also found it beneficial to multiply, after the optimization of w_{\max} and c , gain g of CrossNet somatic cells by a factor of $1/(1 - q) \geq 1$ in comparison with that of the precursor network. According to the first term of Equations (8), this multiplication cancels the defect impact on the *average* amplitude of the post-synaptic signals (though not their random fluctuations due to the defects).

The simplest way to optimize w_{\max} and c is to minimize the r.m.s. perturbation R of weights due to their clipping and crosspoint device defects. Since the values of R for the two weight layers of the network may be quite different, we have used w_{\max} to minimize this parameter for the synapses between the input and hidden layer, and then c to minimize it for the synapses connecting the hidden and output layer. Note that this optimization method involves only the training set (for the adjustment of the continuous synaptic weights during its training), while the test set is only used for the evaluation of the optimized results.

Figure 9(a) shows the results of such optimization,^{††} namely the fraction of classification (test-set) errors for 10 experiments with different random defect positions. (Error bars correspond to the standard deviation on this statistical set.) The left end of the graph (for $q = 0$) shows that the modest weight clipping (with optimized w_{\max}) virtually does not affect the generalization performance of the network: even for a modest level number (say, $L = 33$), the fraction of classification errors ($\sim 2\%$) is close to that in the continuous-weight network, which is, in turn, close to the best results reported in Reference [38]. The growing number of defects, naturally, results in an increase of the number of errors. For example, for the parameters discussed above ($L = 33$, $q = 20\%$) the optimized total error reaches $\sim 4\%$ which is close to the crude estimate given by Equation (15).^{‡‡‡}

In order to check the validity of this simple optimization procedure, we have calculated the classification error on the full test set in the broad range of parameters w_{\max} , different for two synaptic layers, and have optimized it by varying these parameters independently. Figure 9(b) shows the comparison of the results of this ‘perfect’ optimization procedure (which cannot be used in real applications, where the test set is not known in advance) with the simple optimization described above, for the case $L = 33$. We can see that the results are quite comparable, i.e. the simple optimization works reasonably well. It may be further improved in the following way. Considering a part of the training set (in our particular case, 10 000 vectors of 60 000) as the validation set, we look for the minimum of the classification error by changing w_{\max} and c in the small vicinity of the values which have minimized R . As Figure 9(b) shows, this additional procedure makes the CrossNets performance even closer to the ultimate limit.

4. DISCRETE-WEIGHT PRECURSOR

The network fidelity may be further improved by importing synaptic weights from a precursor network with discrete weights, because in this case the precursor training procedure is affected by the weight discreteness, and the weights are not clipped at their import into the CrossNet. This

^{††}The resulting ratios w_{\max}/σ , where σ^2 is the variance of the post-training weight distribution, are close to the values of μ optimal for simple perceptrons (Figure 8(b)), for the same L and q .

^{‡‡‡}The dashed line in Figure 9(a) shows the error in the *precursor* network (with continuous weights) as a function of the fraction of bad synapses. It may seem counter-intuitive that this error is larger than that after weight clipping, for the same q . However, this is readily explained by the fact that the meaning of q for the precursor is different: this is the fraction of bad *synapses* rather than *binary latching switches*.

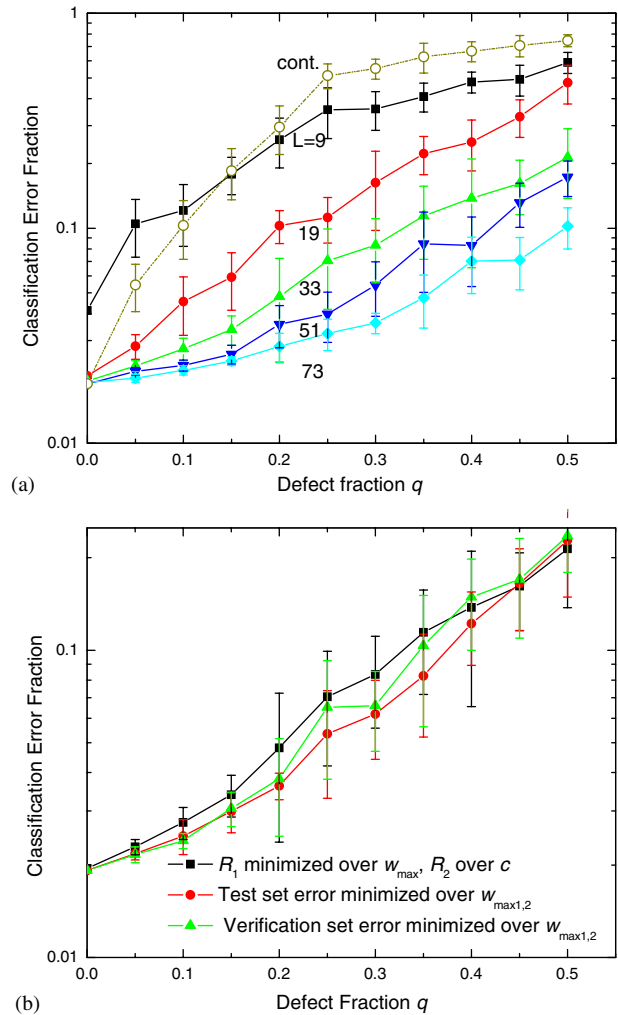


Figure 9. The optimized probability of the classification error of a 784-784-10 CrossNet perceptron (measured on the test set of the MNIST database) as a function of fraction q of bad latching switches: (a) minimization of the error parameter R of the first synaptic layer over the quantized weight amplitude w_{\min} , and R of the second layer over the weight scale c , for several values of L and (b) comparison of the above results with those of the direct minimization of the test set error and of the training subset error over $w_{\max 1,2}$ (for $L = 33$ only). The dashed line on panel (a) shows the error of the precursor network as a function of the fraction of bad *synapses* (rather than *binary latching switches* as for other curves).

idea (usually called ‘soft weight sharing’) was studied earlier, for example, as a means to reduce the necessary synaptic weight memory—see. e.g. the impressive results in Reference [39].

As has been shown in Reference [40], the soft weight sharing may be achieved by the usual error backpropagation, but with additional terms into the cost function. (These terms are minimized when synaptic weights condense to discrete values.) However, this approach demands much more

computing power and training time than usual backpropagation, so that it is hardly suitable for such large networks as CrossNets.

A more practicable approach has been suggested by Shoemaker *et al.* [41]. In their study, weights are initially discrete (e.g. integer), while their updates are restricted to just three values and are made using the following simple rule:

$$\Delta w_{ij} = \begin{cases} +1 & \text{if } x_i \delta_j > +\Delta_0 \\ -1 & \text{if } x_i \delta_j < -\Delta_0 \\ 0 & \text{otherwise} \end{cases} \quad (16)$$

where δ_j are continuous error signals (formed just in the usual backpropagation method), and Δ_0 is a certain prefixed threshold value. It is evident that in this the weights stay discrete (integer).

A disadvantage of this method is the discrete character of the update, i.e. its insensitivity to the exact value of the $x_i \delta_j$ product. Encouraged by the success of our recent work [42] on *in situ* training (to be discussed in Section 5), we have modified the rule (16) by making the updates stochastic. In this approach, the weight update is performed randomly, with the following probabilities:

$$\begin{aligned} \Pr\left(\Delta w_{ij} = +\frac{w_{\max}}{n^2}\right) &\equiv P_1 = \Theta(\zeta_{ij})|\zeta_{ij}| \\ \Pr\left(\Delta w_{ij} = -\frac{w_{\max}}{n^2}\right) &\equiv P_2 = \Theta(-\zeta_{ij})|\zeta_{ij}| \\ \Pr(\Delta w_{ij} = 0) &= 1 - P_1 - P_2 \end{aligned} \quad (17)$$

where

$$\zeta_{ij} \equiv \frac{x_i}{x_{\max}} \frac{\delta_j}{\delta_{\max}} \quad (18)$$

and $\Theta(\zeta)$ is the Heaviside step function (equal to 0 at $\zeta < 0$, and +1 otherwise). Note that in contrast to Equation (16), the probabilities are smooth functions of the product $x_i \delta_j$.

Since CrossNets have limited amplitude of weights (w_{\max}), in order to avoid weight clipping at import, Equation (17) has to be complemented with some rule forbidding the weights to grow beyond the ‘hard walls’ at $\pm w_{\max}$. We have found that the best results are given by the ‘sticky wall’ rule: once the weight has reached one of the hard walls, it ‘sticks’ to it and cannot be further changed, no matter what Equation (17) says.

We tested this training method for the same MNIST classification task using an MLP of the same size (784-784-10) as had been used for the continuous weight import. In this numerical study, three different practicable cases ($L = 33, 51$ and 73) have been explored. The value x_{\max} participating in Equation (18) was the activation function saturation level, while the maximum level δ_{\max} of the back-propagated error, required by that equation, was fixed at $\sqrt{10} \varepsilon_{\max} w_{\max}/2$ for the first layer of synapses, and at ε_{\max} for the second layer. Here ε_{\max} is the maximum level of the output layer error, $w_{\max}/2$ is the expected average backpropagation gain, while the factor $\sqrt{10}$ reflects the number of output stages (10) contributing to the error signal in the first layer.

After training the precursor using the described rule, the discrete weights have been imported into the simulated CrossNet with the same random defect model as for the continuous-weight case (Sections 2 and 3). Figure 10 shows the resulting classification performance of the network with

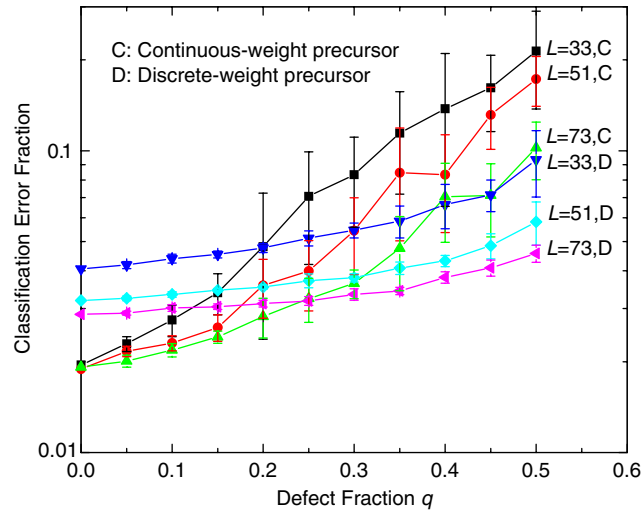


Figure 10. The MNIST set generalization error of CrossNets with weights imported from a discrete-weight precursor networks as a function of the fraction of bad nanodevices, in comparison with that for the continuous-weight precursor results (also shown in Figure 8(a)).

imported weights from both discrete-weight and continuous-weight precursors. The comparison shows that the continuous-weight precursor outperforms the discrete-weight one at lower defect rates, but the discrete precursor training provides better defect tolerance, with better fidelity at the number of defects above 20%.

One more advantage of the discrete precursor training approach is that it may be implemented more easily using nanoelectronic hardware. This opportunity is to be discussed in Section 6.

5. *IN SITU* TRAINING

The synaptic weight import may become impracticable for very large networks, because the operation of the ‘precursor’ network calculating the weights (e.g. implemented as an algorithm running on a digital computer) may be forbiddingly slow. Hence it is important to carry out the synaptic weight adjustment, necessary for the supervised training or unsupervised learning of the networks, ‘*in situ*’, i.e. inside the CrossNets. Though the adjustment algorithms differ broadly [27–29], virtually all of them (including the generic Hebb rule, error backpropagation, global reinforcement, etc.) require implementing the synaptic weight change Δw_{ij} proportional to the product of two analog signals. As has been demonstrated in the previous section, at least some of these algorithms may be modified for discrete weights and hence implemented inside CrossNets, with their discrete-weight synapses.

For example, the rule given by Equations (17)–(18) may be implemented using the stochastic multiplication [42].*** For the reader’s convenience, we briefly describe that idea here—see

***A somewhat similar idea, but in application to digital rather than analog signals, had been discussed in Reference [43].

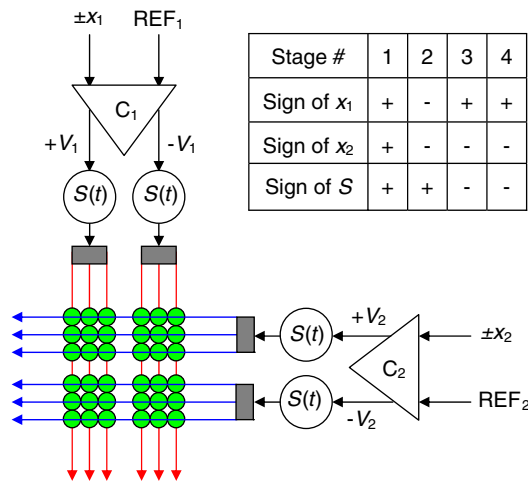


Figure 11. *In situ* training of a CrossNet [42]. Reference signals $REF_{1,2}$ are either random analog signals or deterministic sawtooth waveforms with different frequencies. Each small circle is the latching switch; the composite synapse consists of four groups of $n \times n$ similar switches. (The figure is for the case $n = 3$). $C_{1,2}$ are the signal comparators with binary output signals. The global shift signal $\pm S_0$, together with flipping the signs of signals $x_{1,2}$, performs time division multiplexing. The inset table shows the alternating signs of signals at various steps of the time multiplexing scheme.

Figure 11. It uses the dual-rail presentation for both pre-synaptic (V_1) and post-synaptic (V_2) signals. Each synapse consists of four groups (arrays) of $n \times n$ elementary latching switches. In the operation mode, the open switches supply currents, proportional to axonic voltages, to the inputs of differential dendritic amplifiers, with the same polarities as shown in Figure 11. As a result, the net synaptic weight is

$$w = w_{\max} N / 2n^2, \quad N = N_{++} + N_{--} - N_{+-} - N_{-+} \quad (19)$$

where each component of N is the number of ON-state switches in each group ($0 \leq N \leq n^2$), so that now w may take any of the $L = 4n^2 + 1$ quantized values.

In the training mode, voltages $V_1(t)$ and $V_2(t)$ are developed by comparators $C_{1,2}$ with binary outputs. The comparators are fed by:

- (i) the analog signals $x_{1,2}$ to be multiplied, with the magnitude limited to a certain range $[0, x_{\max}]$, and the signs changed in time to perform 4-stage time division multiplexing (see the table inset in Figure 11); and
- (ii) analog reference signals $REF_{1,2}$ from two independent random signal generators, with the uniform probability within the same range $[0, x_{\max}]$.^{†††}

If the input signal $\pm x_i$ ($i = 1, 2$) is larger than random reference REF_i , the comparator generates voltage $V_i \cong (V_{\text{th}}/2) \text{sgn}(\pm x_i)$, and applies it, in the dual-rail format, to its outputs; otherwise no

^{†††}Each of the signals may be shared by all cells of the same layer, without performance degradation [42].

output is produced.^{†††} A global shift signal $S(t) = \pm S_0$, with an amplitude S_0 somewhat below $V_{th}/2$, and the sign alternating in accordance with the table, is added to the output voltages. It is easy to see that the simultaneous switching of the signs of x_i and $S(t)$ ensures that the net voltage may exceed V_{th} , and hence cause switching with some finite rate (probability per unit time) Γ_0 , only in one group of synapses.

Let us consider, for example, stage 1 for the case when both x_1 and x_2 are positive. In this case only the top left array of switches may be activated, and indeed is if each x_i is larger than REF_i . Due to the uniform probability distribution of each REF signal, the probability that each comparator generates the finite output signal is $P_i = x_i/x_{max}$. Since the signals REF_i are independent, the probability that both comparators produce the finite output, i.e. that the switching rate is finite is

$$P_\Gamma = P_1 P_2 = x_1 x_2 / x_{max}^2 \quad (20)$$

The real law of the average synaptic weight change is slightly different, because the change of probability p of having any latching switch in ON state depends not only on Γ_\pm , but also on p itself

$$dp/dt = \Gamma_+(1-p) + \Gamma_-p \quad (21)$$

Combining Equations (19)–(21), and taking into account all four stages of the time division multiplexing period, we get the following final formula for the average synaptic weight change during a relatively short time interval $\Delta t \ll 1/\Gamma_0$ ^{§§§}:

$$\langle \Delta w \rangle = \eta x_1 x_2 \times \begin{cases} (w_{max} - w), & x_1 x_2 > 0 \\ (w_{max} + w), & x_1 x_2 < 0 \end{cases} \quad (22)$$

with $\eta \propto \Gamma_0 \Delta t$. Note the relaxation term in the right-hand part of (22), giving a contribution into $\langle \Delta w \rangle$, proportional to $-w$; it plays the ‘finite memory time’ role to some extent similar to that in the renowned Oja rule [28].^{¶¶¶}

We have first explored the defect tolerance of this *in situ* method for training a CrossNet-based MLP to perform classification of patterns of three data sets (‘cancer1’, ‘card1’, and ‘diabetes1’) from the popular but relatively small benchmark set Proben1 [44]. The input layer had the number of cells equal to the input vector size (9 for ‘cancer1’, 51 for ‘card1’, and 8 for ‘diabetes1’). The network also had one hidden layer of 10 cells, and two output cells. Each output represented one class, and the Winner Takes All rule was used to define the input vector class. Before the training, all switches were initialized to be randomly ON or OFF with 50% probability. During training, synapses were updated after presenting each pattern of the training set. An elementary

^{†††}Here we assume, for the sake of notation simplicity, that the switching threshold voltages of the latching switches (Figure 1(b)) are similar: $V_+ = V_{th}$, $V_- = -V_{th}$. The generalization of the formulas to the general case, using dc offset $V_{of} = (V_+ + V_-)/2$, is trivial.

^{§§§}This condition means that we are actively exploiting the random character of nanodevice switching. In our simulations described below, the product $\Gamma_0 \Delta t$ was 4×10^{-3} . Still, the interval Δt should include at least one full time division multiplexing period. In our simulations, the period was equal to Δt .

^{¶¶¶}The hardware complexity may be reduced using two periodic sawtooth waveforms with different periods τ_i , instead of analog random signals, as the reference signals REF_i , within the same general scheme, which provides comparable performance [42].

Table I. Classification (test set) error for the case of perfect nanodevices ($q = 0$).

Problem	<i>In situ</i> backpropagation for $n = 4$	Continuous backpropagation*
cancer1	0.010 ± 0.004	0.011
card1	0.14 ± 0.01	0.14
diabetes1	0.26 ± 0.02	0.25

*The results for the continuous backprop training are from Reference [45] which gives only the best run results (no statistics).

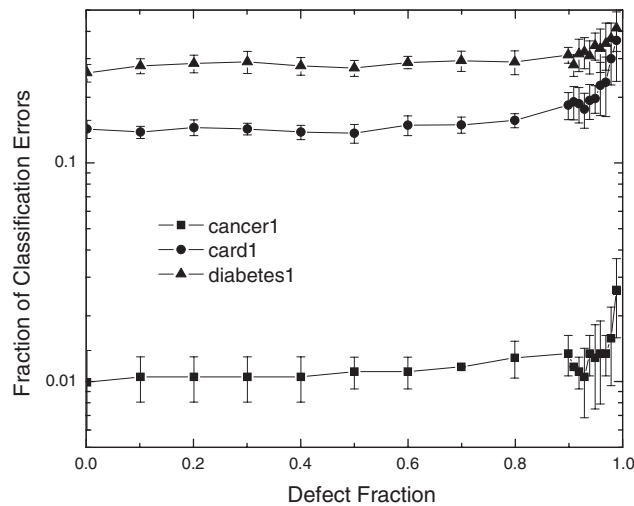


Figure 12. The generalization error of an *in situ*-trained CrossNet for the Proben1 benchmark problems, as a function of the fraction of device defects (introduced before training).

signal preprocessing—pattern centring—has been used

$$(x_j^p)_{\text{centred}} = x_j^p - \frac{1}{P} \sum_{\mu=1}^P x_j^\mu \quad (23)$$

where P is the number of data vectors in the training set. (For centring test patterns, the averaging is still carried out over the training set.) This procedure has improved results for ‘diabetes1’ rather significantly. For other data sets, the initial vectors are already virtually centred, so that the preprocessing has improved the results only marginally. The cell gain g has been optimized for the best performance.

Table I summarizes our best results for $n = 4$ ($L = 4 \times 4^2 + 1 = 65$). It shows that for all the three problems they are at least on a par (and actually better) than those reported in literature for software-implemented networks with deterministic, continuous synaptic weights [45]. Another formulation of the same statement is that synaptic weight discreteness effects are not discernable, at least at this level ($n = 4$).

For the defect tolerance study, we have removed random latching switches, with probability q , from the synaptic arrays before the *in situ* training. Figure 12 shows the result of this procedure

for the same network whose performance is listed in the first column of Table I. One can see that the error does not change noticeably when the fraction of bad nanodevices is below 50%, and starts to grow substantially only when q is well above 90% (!). A possible interpretation of such extraordinary defect tolerance is that at *in situ* training the synaptic weights automatically self-adjust to compensate the fluctuations of the number of available (good) latching switches in each synapse.

Unfortunately, so far we have not been able to make this method work well for a large problem like the MNIST data set. The classification error of CrossNets trained with this algorithm is about 14% even for large MLPs (e.g. 784-500-10) with 36 switches in each synapse trained with 10 000 (out of 60 000) training samples. This error is 10% higher than that of a continuous-weight MLP of the same size, trained with 10 000 training samples, and comparable to linear networks performance (12%) trained with full training set [36]. Although more hidden neurons and switches can improve the performance, it is hard to test it due to demanding computing power. Based on current performance, it is disappointing because discrete precursor training performances with 32 switches show comparable results to that of continuous backpropagation.

6. MIXED-MODE TRAINING

The lower learning ability can be explained by the fact that Equation (22) has the relaxation term which prevents synaptic weights from reaching and keeping values near either end of their range $[-w_{\max}, +w_{\max}]$. This gave us a motivation to explore a *mixed-mode* implementation of the rule given by Equations (17) and (18), at which the (discrete) synaptic weights are being stored in CrossNet synapses, but their adjustment is carried out by an external system. If the adjustment is done sequentially, or at least not completely in parallel, this approach may substantially reduce the required external hardware resources, though the training time may be larger than at the completely *in situ* approach.

Indeed, let us replace the synaptic arrays shown in Figure 11 with the usual, CMOS-implemented AND gates. Since the probability of having two random binary inputs of the gate equal 1 simultaneously is proportional to the product of probabilities of each of the events, the probability of having 1 at the gate output will be proportional to $|x_1 x_2|$. We now can read out the current weight of the CMOL synapse, update the weight in accordance with Equations (17) and (18) (complemented with the 'sticky wall' rule), and write the resulting new value of w_{ij} back to the synapse.

In the limit of perfect hardware the resulting training algorithm (which we call Method 1) is exactly the same as that for the discrete precursor training (Section 4). However, since the mixed-mode training involves the external measurement of each external weight (including the effect of bad switches), it may be optimized to provide higher defect tolerance. The most apparent approach here is to check the result of the weight change, and if it has not led to the desired result (this would happen if the switch whose state is being changed is defective), repeat the procedure with a different binary switch until the desired weight value has been reached. In the numerical simulations, higher normalized gain g has been used in networks with more defective switches in order to keep the same level of non-linearity

$$g = 2\sqrt{\frac{3}{784}} \frac{1}{1-q} \quad (24)$$

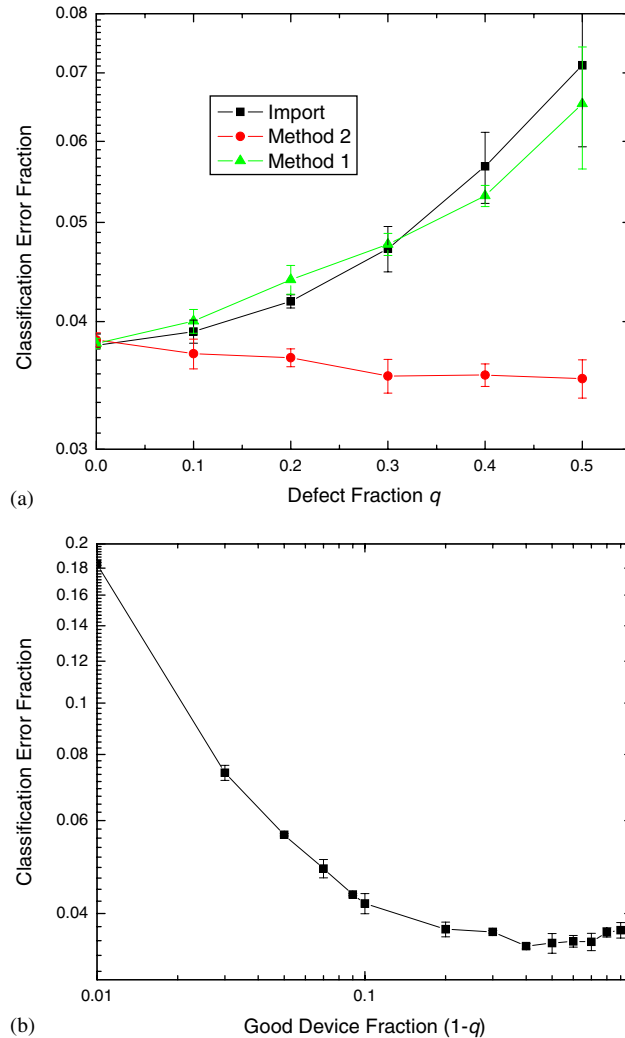


Figure 13. (a) The generalization error for two methods of the mixed-mode training, for the MNIST benchmark set, at $L=33$. For comparison, the discrete-weight import results for the same task and network (also presented in Figure 10) are also shown and (b) the generalization errors of Method 2 for the higher defect rates. (In this region, fewer experiments have been conducted: for $q>90\%$, three experiments, and for $60\%<q<90\%$, only two experiments.)

The results of the classification performance calculation for Method 1, using the MLP of the same size (784-784-10) and 32 switches for each synapse, are shown in Figure 13(a). The figure shows that fidelity provided by this method is close to that reached by the discrete-weight import, slightly outperforming it at higher defect fraction, and hence is inferior to that of the continuous weight import at small q (Figure 10). This may be explained by the competition between increasing

defect tolerance and reducing the effective number of quantized levels of synaptic weights: for defective synapses ($q \neq 0$), the maximum number of binary switches usable in each array becomes less than n^2 , and hence the maximum number of useful levels of the synaptic weight becomes lower than $L = 2n^2 + 1$.

We have been able to reach much better defect tolerance with an alternative Method 2 in which the desired synaptic weights are stored in CMOS memory and imported to the synapses of CrossNets without any error correction, thus allowing CrossNets synapses to have values different from the ‘perfect’ weights stored in the external memory. Results of this procedure, obtained for the same task with the same network and with the same gain (24), are also presented in Figure 13(a). Counter-intuitively, a modest density of defects provides *better* performance than the perfect network, and only at very high defect fraction the fidelity is degraded (Figure 13(b)). Right now, we do not have a good explanation for this behaviour. (Our plans are to explore this issue.)

The price for this extraordinary defect tolerance of CrossNet trained by Method 2 is the necessity to use an external system for storing synaptic weights in the part of CrossNet chip which is currently activated for training. On the other hand, at Method 2 all the synapses connected to the same somatic cell and having equal current synaptic weights and calculated adjustments can be updated in parallel. As a result, the number of necessary update steps is proportional to the product of L by the number of neurons in the layer.

Since it is possible to implement the rule given by Equations (17) and (18) with CMOS circuits only, it may be interesting to carry out a comparison between their CMOL and CMOS implementations. A single CMOS latching switch may be realized by one SRAM cell and one pass transistor with the gate connected to the bit line of SRAM cell. (In order to realize the linear conductance of a latching switch, the pass transistor should work in the non-saturation region.) According to this simple model, 224 transistors are necessary for a 32 quantized level synapse. This means that processing of a very small (by present-day standards) image from 1000 pixels with a simple perceptron (1000–1000) would require above 200 million transistors for synapses only, i.e. a CMOS circuit comparable to Intel’s Core2 Duo CPU. On the other hand, the similar synapse matrix implemented with CrossNet with the plausible half-pitch $F_{\text{nano}} = 4$ nm would take just $32 \times 64 \mu\text{m}^2$, and a 1-cm²-scale CrossNets can provide very fast classification of megapixel-scale images, especially if quasi-local rather than global connectivity is required [31].

7. DISCUSSION

For practical applications of neuromorphic CMOL networks, our main results are very encouraging: the feedforward CrossNets trained as pattern classifiers by all three considered methods (weight import, *in situ* training when practicable, and mixed-mode training) has turned out to be very insensitive to both synaptic weight discreteness and nanodevice defects (at least those equivalent to the ‘stuck-at-open’ faults). For example, at the import from a continuous-weight precursor, the use of 4×4 arrays of binary latches instead of the continuous synapses does not produce any discernable negative effects on the classification performance. Of these nanodevices, up to 20% may be defective before the classification error increases by a factor of two—see Figure 9.

Even higher defect tolerance may be obtained at weight import from a precursor network with discrete synapses (Figure 10). Though the classification error at such import is somewhat higher than in the previous case if the fraction q of bad devices is low, this error does not increase substantially up to $q \sim 50\%$. Moreover, virtually similar results may be reached using the mixed-

mode training in which CrossNet is used for training signal propagation and hence the external training system may be substantially smaller.

This does not mean, however, that the problem does not require further attention. On one hand, the hard defects of the opposite type (crosspoint shorts, to some extent equivalent to ‘stuck-at-close’ transient faults) may be more damaging, because they may invalid not just a single crosspoint, but two involved nanowire fragments connected to many ($2M$) crosspoint devices. A quantitative analysis of this problem is an urgent task.

On the other hand, there are considerable reserves for increasing the defect tolerance. Indeed, CMOL topology allows one to test every crosspoint device and, if necessary, replace nanowire fragments with the largest number of bad devices for spares. For CMOL digital logic [25] and memory [24] circuits, this procedure has proven to be very effective. We have not yet tried to carry out a similar procedure for CrossNets, where it would be most natural at weight import.

Finally, the clipping and defect tolerances have to be explored on the level of large-scale real-world classification problems like that analysed in Reference [31]. Our plans include pursuing these goals.

ACKNOWLEDGEMENTS

Useful discussions with P. Adams, J. Barhen, D. Hammerstrom, P. Werbos, and especially X. Ma are gratefully acknowledged. The work has been supported in part by AFOSR, MARCO *via* FENA Center, and NSF.

REFERENCES

1. *International Technology Roadmap for Semiconductors. 2006 Edition*. Available online at <http://public.itrs.net/>
2. Likharev KK. Electronics below 10nm. In *Nano and Giga Challenges in Microelectronics*, Greer J *et al.* (eds). Elsevier: Amsterdam, 2003; 27–68.
3. Kuekes PJ, Snider GS, Williams RS. Crossbar nanocomputers. *Scientific American* 2005; **293**(11):72–77.
4. DeHon A, Likharev KK. Hybrid CMOS/nanoelectronic digital circuits. *Proceedings of ICCAD'05*. IEEE: Piscataway, NJ, 2005; 375–382.
5. Chen A, Haddad S, Wu Y-C, Fang T-N, Lan Z, Avanzino S, Pangrle S, Buynoski M, Rathor M, Cai W, Tripsas N, Bill C, VanBuskirk M, Taguchi M. Non-volatile resistive switching for advanced memory applications. *IEDM'05 Technical Digest*. IEEE: Piscataway, NJ, 2005; 31.3.
6. Baek IG, Kim DC, Lee MJ, Kim H-J, Yim EK, Lee MS, Lee JE, Ahn SE, Seo S, Lee JH, Park JC, Cha YK, Park SO, Kim HS, Yoo IK, Chung U-In, Moon JT, Ryu BI. Multi-layer cross-point binary-oxide resistive memory (OxRAM) for post-NAND storage applications. *IEDM'05 Technical Digest*. IEEE: Piscataway, NJ, 2005; 31.4.
7. Ma LP, Xu Q, Yang Y. Organic electrical bistable devices and rewritable memory cells. *Applied Physics Letters* 2002; **80**(16):4908–4910.
8. Bozano LD, Kean BW, Deline VR, Salem JR, Scott JC. Mechanism for bistability in organic memory elements. *Applied Physics Letters* 2004; **84**(4):607–609.
9. Sezi R, Walter A, Engl R, Maltenberger A, Schumann J, Kund M, Dehm C. Organic materials for high-density non-volatile memory applications. *IEDM'03 Technical Digest*. IEEE: Piscataway, NJ, 2003; 10.2.1.
10. Lai Y-S, Tu C-H, Kwong D-L, Chen J-S. Bistable resistance switching of poly(*N*-vinylcarbazole) films for nonvolatile memory applications. *Applied Physics Letters* 2005; **87**(12):122101.
11. Collier CP, Wong EW, Belohradsky M, Raymo FM, Stoddart JF, Kuekes PJ, Williams RS, Heath JR. Electronically configurable molecular-based logic gates. *Science* 1999; **285**(5426):391–394.
12. Li C, Zhang D, Li X, Han S, Tang T, Zhou C, Fan W, Koehne J, Han J, Meyyappan M, Rawlett AM, Price DW, Tour JM. Fabrication approach for molecular memory arrays. *Applied Physics Letters* 2003; **82**(4):645–647.
13. Wu W, Jung GY, Olynick DL, Straznicky J, Li Z, Li X, Ohlberg DAA, Chen Y, Wang S-Y, Liddle JA, Tong WM, Williams RS. One-kilobit cross-bar molecular memory circuits at 30nm half-pitch fabricated by nanoimprint lithography. *Applied Physics A* 2005; **80**(6):1173–1178.

14. Chen Y-C, Chen F, Chen CT, Yu JY, Wu S, Lung SL, Liu R, Lu C-Y. An access-transistor-free (0T/1R) non-volatile resistance random access memory (RRAM) using a novel threshold switching, self-rectifying chalcogenide device. *IEDM'03 Technical Digest*. IEEE: Piscataway, NJ, U.S.A., 2003; 37.4.1.
15. Kund M, Beitel G, Pinnow C-U, Röhr T, Schumann J, Symanczyk R, Ufert K-D, Müller G. Conductive bridging RAM (CBRAM). *IEDM'05 Technical Digest*. IEEE: Piscataway, NJ, 2005; 31.5.
16. Likharev KK, Mayr A, Muckra I, Türel Ö. CrossNets: high-performance neuromorphic architectures for CMOL circuits. *Annals of the New York Academy of Sciences* 2003; **1006**:146–163.
17. Fölling S, Türel Ö, Likharev KK. Single-electron latching switches as nanoscale synapses. *Proceedings of the 2001 IJCNN*. International Neural Network Society: Mount Royal, NY, 2001; 216–220.
18. Dresselhaus PD, Ji L, Han S, Lukens JE, Likharev KK. Measurement of single-electron lifetimes in multijunction traps. *Physical Review Letters* 1994; **72**(20):3226–3229.
19. Ji L, Dresselhaus PD, Han SY, Lin K, Zheng W, Lukens JE. Fabrication and characterization of single-electron transistors and traps. *Journal of Vacuum Science and Technology B* 1994; **12**(6):3619–3622.
20. Park J, Pasupathy AN, Goldsmith JI, Chang C, Yaish Y, Petta JR, Rinkoski M, Sethna JP, Abruna HD, McEuen PL, Ralph DC. Coulomb blockade and the Kondo effect in single-atom transistors. *Nature* 2002; **417**(6890):722–725.
21. Kubatkin S, Danilov A, Hjort M, Cornil J, Bredas JL, Stuhr-Hansen N, Hedegard P, Bjornholm T. Single-electron transistor of a single organic molecule with access to several redox states. *Nature* 2003; **425**(6959):698–701.
22. Zhitenev NB, Jiang WR, Erbe A, Bao Z, Garfunkel E, Tennant DM, Cirelli RA. Control of topography, stress and diffusion at molecule–metal interfaces. *Nanotechnology* 2006; **17**(5):1272–1277.
23. Strukov DB, Likharev KK. CMOL: devices, circuits, and architectures. In *Introducing Molecular Electronics*, Cuniberti G *et al.* (eds). Springer: Berlin, 2005; 447–477.
24. Strukov DB, Likharev KK. Architectures for defect-tolerant nanoelectronic crossbar memories. *Journal of Nanoscience and Nanotechnology* 2007; **7**(1):151–167.
25. Strukov DB, Likharev KK. A reconfigurable architecture for hybrid CMOS/nanodevice circuits. *Proceedings of the FPGA'06*. ACM: New York, 2006; 131–140.
26. Türel Ö, Lee JH, Ma X, Likharev KK. Neuromorphic architectures for nanoelectronic circuits. *International Journal of Circuit Theory and Applications* 2004; **32**(5):277–302.
27. Widrow B, Lehr MA. 30 years of adaptive neural networks. *Proceedings of the IEEE* 1990; **78**(9):1415–1442.
28. Hertz J, Krogh, Palmer RG. *Introduction to the Theory of Neural Computation*. Perseus: Cambridge, MA, 1991.
29. Bishop CM. *Neural Networks for Pattern Recognition*. Oxford University Press: Oxford, U.K., 1995.
30. Mountcastle VB. *The Cerebral Cortex*. Harvard University Press: Cambridge, MA, 1998.
31. Lee JH, Likharev KK. CrossNets as pattern classifiers. *Lecture Notes in Computer Science* 2005; **3512**:446–454.
32. Xie Y, Jarbi MA. Analysis of the effects of quantization in multilayer neural networks using a statistical model. *IEEE Transactions on Neural Networks* 1992; **3**(2):334–338.
33. Dündar G, Rose K. The effects of quantization on multilayer neural networks. *IEEE Transactions on Neural Networks* 1995; **6**(6):1446–1451.
34. Snider G, Kuekes P, Williams RS. CMOS-like logic in defective, nanoscale crossbars. *Nanotechnology* 2004; **15**(8):881–891.
35. DeHon A, Naeimi H. Seven strategies for tolerating highly defective fabrication. *IEEE Design and Test of Computers* 2005; **22**(4):306–315.
36. LeCun Y, Bottou L, Bengio Y, Haffner P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE* 1998; **86**(11):2278–2324.
37. LeCun Y, Cortes C. *The MNIST Database of Handwritten Characters*. Available online at <http://yann.lecun.com/exdb/mnist/>
38. Bhattacharya U, Vijda S, Mallick A, Chaudhari BB, Belaid A. On the choice of training set, architecture and combination rule of multiple MLP classifiers for multi-resolution recognition of handwritten characters. *Proceedings of the 9th International Workshop on Frontiers in Handwriting Recognition (IWFHR-9)*, 2004; 419–424.
39. Köksal F, Alpaydin E, Dündar G. Weight quantization for multi-layer perceptrons using soft weight sharing. *Lecture Notes in Computer Science* 2001; **2130**:211–216.
40. Nowlan SJ, Hinton GE. Simplifying neural networks by soft weight-sharing. *Neural Computation* 1992; **4**(4): 473–493.
41. Shoemaker PA, Carlin MJ, Shimabukuro RL. Backpropagation learning with trinary quantization of weight. *Neural Networks* 1991; **4**(2):231–241.
42. Lee JH, Likharev KK. In situ training of CMOL CrossNets. *Proceedings of the WCCI/IJCNN'06*, 2006; 5026–5034.

43. Kondo Y, Sawada Y. Functional abilities of a stochastic logic neural network. *IEEE Transactions on Neural Networks* 1992; **3**(5):434–443.
44. Prechelt L. Proben1—a set of neural network benchmark problems and benchmarking rules. *Technical Report 21/94*, Fakultät für Informatik, U. Karlsruhe, Karlsruhe, Germany, 1994.
45. Prechelt L. Automatic early stopping using cross validation: quantifying the criteria. *Neural Networks* 1998; **11**(4):761–767.
46. Akkerman HB, Blom PWM, de Leeuw DM, de Boer B. Towards molecular electronics with large-area molecular junctions. *Nature* 2006; **441**(7089):69–72.