An Analog Processor Architecture for a Neural Network Classifier

Many neural-like algorithms currently under study support classification tasks. Several of these algorithms base their functionality on LVQ-like procedures to find locations of centroids in the data space, and on kernel (or radial-basis) functions centered on these centroids to approximate functions or probability densities. A generic analog chip could implement in a parallel way all basic functions found in these algorithms, permitting construction of a fast, portable classification system.

Michel Verleysen

Philippe Thissen

Jean-Luc Voz

Catholic University of Louvain

Jordi Madrenas

Polytechnic University of Catalonia

Nonparametric classification of data is certainly one of the main tasks that let us successfully compare neural networks, or neural-like algorithms, to the more conventional methods known in information theory. However, these algorithms generally require large amounts of computation, and some applications also require real-time, portable classification systems that do not continuously connect to a host computer. We set out to design a fast, portable system to fill each of these requirements.

Our design of a parallel analog implementation of a VLSI processor efficiently computes the operations involved in most neural-like classification algorithms and implements the recognition phase of classification tasks. It assumes that an external host computer has achieved learning, and that the results of this learning have been downloaded on the analog processor through the digital control unit.

Algorithms for classification tasks

Classification algorithms generally behave as follows. First, the algorithms partition into classes a multidimensional space containing data, which usually measures in a physical system. The problem itself generally determines the number of these classes, while the algorithms compute the shape of each partition in the space according to the distribution of stimuli or input data. Distribution, of course, includes the class label associated with each piece of input data during (supervised) learning. Then, when the partitions are fixed, the algorithms can classify each new input data, without class labels, by determining to which partition it belongs.

Classifiers support various application domains such as signal processing (image compression, phoneme or sound recognition), optical character recognition, detection of anomalies in fabrication lines, and detection of abnormal conditions in an industrial process or a huge number of measurements. Usually, the supervised learning and generalization phases—the periods during which partitions are determined and vectors are classified—are completely separated. In some practical situations, however, these two phases overlap. Then we use an adaptive algorithm to adapt the shapes and sizes of the partitions as new data belonging to known classes appear. This approach avoids the need to recompute the whole set of parameters, which would require the memorization of all input data since the beginning of the learning process.

To illustrate the concept of data classification, we introduce two kinds of algorithms: learning (or adaptive) vector quantization (LVQ) and kernel-based classifiers of probability densities (KBC). Depending on the complexity of the prob-

0272-1732/94/$04.00 © 1994 IEEE

lem, dimension of the data space, number of classes, overlap between classes in the space, decision to or not to obtain a Bayesian classifier, speed requirements, and many other factors, we can use either method. Researchers are currently studying both methods in many different areas requiring classification. We describe them here as "case studies;" they are not the only way to build classifiers. We do not evaluate the respective advantages and drawbacks of these methods.

LVQ algorithm. Vector quantization finds in the input space a restricted number of patterns, called centroids, whose distribution (probability density) is as close as possible to the whole set of input vectors. In other words, vector quantization restricts the number of data points for further processing, keeping as much information as possible from the initial distribution.

Kohonen[1] proposed several LVQ algorithms. We discuss the original LVQ1 algorithm; others are based on the same principle of moving centroids depending on input data and require the same computing resources as the LVQ1. Consider in the following that N d-dimensional input vectors $x_j (1 \le j \le N)$ form the input distribution, and that there are P centroids (or prototypes) p_i $(1 \le i \le P)$. We can describe the LVQ1 algorithm then as follows.

Before the first iteration, the algorithm randomly initializes the P prototypes p_i. If a priori limits on the set of input vectors are known, LVQ1 chooses the prototypes inside these limits; one possibility is to initialize the prototypes to any P of the N input vectors. At each iteration, LVQ1 compares a d-dimensional input vector x_j to all prototypes and selects the one p_a for which the standard Euclidean distance between p_a and x_j is minimal, according to

$$\|p_a - x_j\| \le \|p_k - x_j\|, \forall k \varepsilon (1 \ldots P) \setminus (a), 1 \le a \le P \qquad (1)$$

If vectors x_j and p_a belong to the same class, p_a moves in the direction of x_j:

$$p_a = p_a + \alpha(x_j - p_a)$$

where α is an adaptation factor $(0 \le \alpha \le 1)$. If the two vectors belong to different classes, p_a moves in the opposite direction.

Adaptation factor α must decrease with time to obtain a good convergence of the algorithm. Usually, the algorithm keeps the same value of α for a whole epoch and decreases it before the next one. (An epoch consists in presenting once the whole set of input patterns.)

After convergence, that is, after several epochs, the algorithm locates centroids p_i in the space so that their distribution represents the initial distribution of input vectors x_j. LVQ1 is an iterative version of the Linde-Buzo-Gray method,[2] which is commonly used in vector quantization. After training, we use Equation 1 to select centroid p_a, which is closer to a new input x_j than any other centroid. Then we attribute the class of p_a to the input pattern to achieve classification. We can adequately approximate the boundaries between classes only if we chose the number of centroids in each class proportional to the respective a priori probability densities of the classes.[1] Thus, we approximate a class a priori more probable than another with more centroids, which seems intuitively reasonable.

KBC algorithm. Kernel-based classifiers work in two phases. They estimate the probability density of the data distribution inside each class. Then, they use the Bayes law to determine the boundaries between classes in the data space, and thus classify new input vectors.

To estimate the probability density of data belonging to a particular class, we sum the kernels centered on the data from the learning set available in this class:

$$\hat{p}_x(N_c, u|w_c) = \frac{1}{N_c} \sum_{i=1}^{N_c} \Phi\left(\frac{u - x_i}{h}\right) \qquad (2)$$

Definitions

Bayes law	Fundamental law for data classification in statistics
Cascode	Commonly used type of cell in analog VLSI chips
Centroids	A restricted number of patterns whose distribution is as close as possible to the distribution of the whole set of input vectors
Gaussian function	Exponentially decreasing function commonly used in statistics
KBC	Kernel-based algorithm that approximates functions or probability densities
Kernel	Radial function centered on a centroid
Learning process	Adaptive process to approximate unknown parameters through examples
LVQ	Adaptive vector quantization algorithm that finds locations of centroids in the data space
Manhattan distance	Cityblock distance
Radial basis	Function depending only on the distance and not on the direction
Synapse	Connection between neurons
Weights	Multiplying factors of synapses
Winner-take-all	Cell that selects the maximum value among several values

where $(x_i, 1 \le i \le N_c)$ denotes the samples at disposal in class w_c. We suppose that there are C classes denoted w_c, $1 \le c \le C$. The scalar parameter h is called the width factor of the kernel. Kernel Φ is said to be radial if it is only a function of the norm of its argument. Several types of Φ kernels may be used, the most classical one being a Gaussian function:

$$\Phi\left(\frac{u-x_i}{h}\right) = \frac{1}{\left(\sqrt{2\pi}\right)^d}\frac{1}{h^d}e^{-\frac{1}{2}\left(\frac{\|u-x_i\|}{h}\right)^2} \tag{3}$$

where d is the dimension of u and x_i.

Cacoullos proves the convergence of such an estimator $\hat{p}_x(N_c, u \mid w_c)$ to the true density $p_x(u \mid w_c)$ in the mean square sense[3,4] when N_c tends to infinity and the kernels Φ respect some conditions.

We can vary width factor h for each kernel $\Phi\ (u - x_i / h)$; in this case we refer to kernels as variable rather than fixed. However, the prohibitive number of operations involved in the case of variable kernels make them rather inefficient in terms of software or hardware implementations. Furthermore, tests show limited gain in performances between estimators based on fixed and variable kernels, especially for finite databases. For these reasons, we consider only fixed-kernel estimators here.

Finally, even if the probability density estimators are shown to be asymptotically unbiased, we can reduce the number of computations in practical applications by reducing the number of samples through some kind of vector quantization, for example, an LVQ procedure. For the LVQ procedure to give an appropriate distribution of the centroids in each class, we must set their number proportional to the a priori probabilities of the respective classes.

Once the probability densities are estimated in each class, we can use the Bayes criterion to classify any new vector x; class w_c $(1 \le c \le C)$ will be attributed to vector x if

$$\hat{p}_x(x|w_c)P(w_c) \ge \hat{p}_x(x|w_i)P(w_i),\ 1 \le i \le C \tag{4}$$

where $P(w_i)$ is the a priori probability of class w_i. Such a classifier is interesting mostly because of its property to approximate the Bayes limits between classes, that is, the boundaries leading to a minimum number of misclassifications in case of overlapping distributions. Most other classification systems do not have this property.

Complexity of the classification algorithms and advantages of a parallel chip. A weak point of the two classes of algorithms certainly resides in the number of operations involved in classifying data (after learning). Indeed, for each input vector to classify, both algorithms require calculation of at least P distances between this input vector and the centroids. After that, the LVQ algorithm selects the minimum of these distances (and computing a minimum of P values necessitates at least $P - 1$ comparisons). Alternatively, KBC divides the resulting distances by the respective width factors to serve as inputs to kernel functions (exponentials), which then must be added and compared. This implies a lot of computations, directly proportional to the number of centroids memorized in the network and to the number of data to classify.

Speed is not always a limitation in applications. When monitoring a process line, for example, we do not need to classify the measurements made on this line with a delay that is much shorter than the time constant of the physical process itself. However, in many other applications, like speech recognition or image compression, the number of data to classify is so huge that it would be difficult to hold real-time processing on a conventional personal computer or workstation, or even on a dedicated signal processor.

We immediately imagined building a specialized VLSI processor to speed up these operations. The nature of the algorithms makes a parallel processor the best candidate for such implementation. For example, $P \times d$ identical cells can implement a distance between P d-dimensional centroids and one d-dimensional input vector by calculating distances on one component of the vectors, their results being summed by groups of d. We propose an architecture that uses analog computation blocks to realize these operations.

Realizing a dedicated chip for classification is not only interesting for its speed. Many industrial applications require a portable stand-alone and easy-to-use classifier. This means that, after learning, the system should be able to indicate the class of a vector, representing both input and output as electrical signals, either analog or digital. In addition, the system should be ready to operate without complex programming. We based our architecture on an analog operative chip, coupled with a digital control part, which fulfills these requirements.

Mixed analog-digital architecture

Specialized VLSI cells can easily implement all the operations we've just described (addition, multiplication, distance computation, nonlinear Gaussian-like functions, winner-take-all operation). Sums, winner-take-all, and nonlinear kernel functions are particularly suited to analog realizations. We can easily realize nonlinear functions, using the nonlinear characteristics of transistors and amplifiers, and sums just by connecting several current sources on common current lines. Various researchers have studied analog winner-take-all functions.

It is more difficult to decide how the values will be memorized. Digital memory points are, of course, the easiest to implement. Static memory points require a large silicon area, and the solution is thus oriented toward dynamic ones (which require refreshment). The problem is still the number of memory points that would be necessary to store on the chip, if one memory point must be used for each bit of each stored pattern. Another drawback is the availability of the data in the chip in digital form, which requires a digital-to-analog conversion before further analog processing.

Even if we use analog cells to carry out most operations in the two classification algorithms, a sequencing of the operations must obviously take place in the architecture. The sequencing determines which operations will be used, in which order, and which input and output lines must be used.

Figure 1 shows the principle of the architecture used to implement the classification algorithms. It consists of two different parts. An analog processor classifies data, by means of one of the algorithms. We essentially restrict inputs and outputs of this part to the classification data, their classes (at the input for memorization and at the output for classification), and some control lines. The control lines include parameters to characterize some operations (the shape of Gaussian kernels, for example) and choices between one operation or another. This part of the architecture is independent of the algorithm used, in the sense that a unique processor is implemented to cover the whole set of operations needed in the different algorithms.

We devoted the second part of the architecture to control. It essentially uses the outputs of the processing part to sequence the operations and the allocation of units. It also provides the control lines necessary to the analog processor. This part is dependent on the algorithm, since different algorithms will need different sequences of operations, and the use of different output lines of the processor. The control section is a digital finite-state machine with either a specialized or general-purpose digital chip, discrete components on a printed-circuit board, or even FPGAs. Combining the analog processor and the digital control section leads to an efficient architecture for classification tasks.

Analog processor

Figure 2, next page, shows that we've built the core of our system around P identical cells. Each cell contains memory points to store the coordinates of the centroid and its class, together with a distance calculator to compute the distance between this centroid and an input vector. Briefly, the system stores a set of P centroids p_k, $1 \leq k \leq P$ in the processor; in the case of the KBC algorithm, the coordinate of the centroid corresponds to the center of kernel function Ω. Then, when an input vector enters the circuit for classification, the algorithms compute all distances between this input vector and each of the centroids in parallel; this is the purpose of the P distance computation cells.

The algorithms use P computed distances in two ways. On one hand, they compare distances to find the smallest one so they can select the closest centroid from the input vector. The LVQ algorithm uses the shortest distance when selecting centroid p_a in Equation 1. On the other hand, the distances serve as inputs to P Gaussian-like kernel functions used in KBC algorithms, as mentioned in Equation 3.

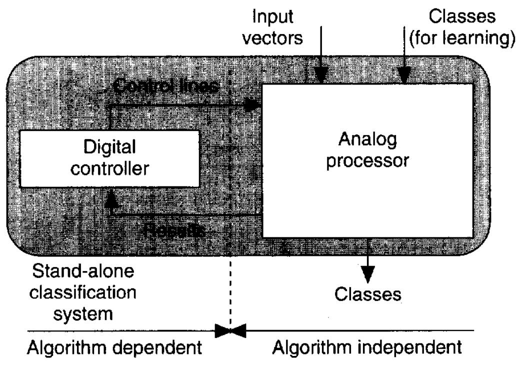


Figure 1. Mixed architecture for classification algorithms.

The LVQ algorithm uses the selection of the winning centroid p_a to complete the recognition phase of the algorithm. The KBC algorithm sums the P kernel outputs class by class, according to Equation 2 to estimate the probability densities of each class. External commands, as detailed later, adjust the parameters of the kernels, namely their widths and shapes. According to Bayes law (Equation 4), the algorithm then classifies the input pattern by selecting the largest probability density from among the different classes.

Equation 4 gives us supplementary factor $P(w_i)$, which corresponds to the a priori probabilities of the classes. However, the same equation supposes that $\hat{p}_x(x \mid w_i)$ represents the estimate of the probability densities, that is, functions whose integral is 1. This is why we multiplied factors in Equation 3.

In our circuit, however, we can adjust the parameters of the kernels themselves. In reality, each kernel in the circuit has an identical integral. Suppose this integral is equal to 1. Though this is pure convention, multiplying all kernel values by a constant does not change the classification decision in Equation 4. Summing all kernels in each class will thus lead to an estimate of each probability density proportional to the respective numbers of kernels. If this number is itself proportional to the a priori probabilities $P(w_i)$ in Equation 4, the maximum value from among probability densities estimates of each class computed by the second winner-take-all in Figure 2 will respect the Bayes decision. Since the number of kernels in each class must already be proportional to the a priori probabilities for a correct boundaries approximation through the LVQ algorithm, the chip will adequately estimate the Bayes boundaries between classes.

The circuit operates as follows. Input vectors, as well as the coordinates of the centroids for storage in the circuit, enter the chip as analog voltages. As seen later, the same input circuitry accommodates one coordinate of the input vector and the corresponding coordinate of the centroid, to

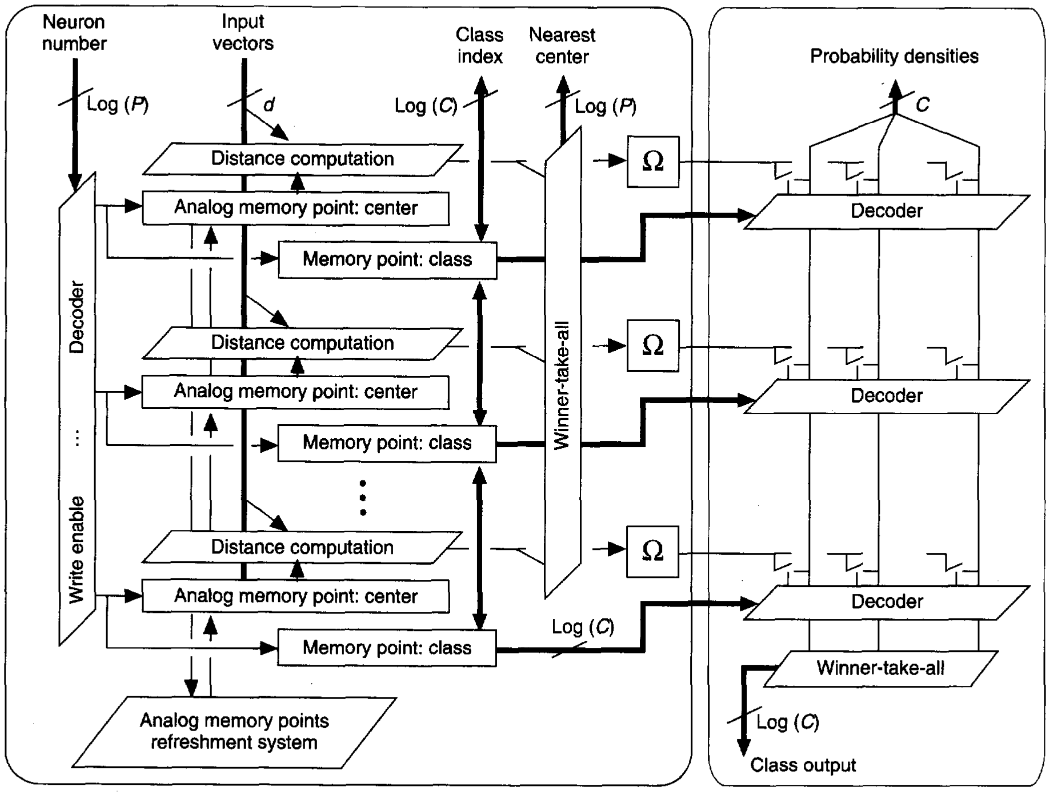


Figure 2. Functional description of the analog processor.

eliminate mismatching problems. Each coordinate of each centroid is stored in an analog memory point; simultaneously, the corresponding class label of the centroid is stored in classical static digital memory points.

During the distance computation, each current corresponding to one coordinate of one centroid will be subtracted from the current representing the corresponding coordinate of the input vector. The algorithm obtains d values in this way for each centroid and sums them on current lines to compute the Manhattan distance. This takes place before entry of the winner-take-all and kernel functions. We justify the use of the Manhattan distance instead of the Euclidean distance later.

Outputs of kernel functions are also current. They are in turn summed on a second set of C current lines, one per class. A set of decoders connected to the static memory points that store the class labels selects the line where a particular current must be added. The second winner-take-all finally selects the class corresponding to the largest estimate of the probability densities (multiplied by the a priori probabilities), to complete the classification process.

We chose the sizes of all transistors to correspond to cells designed in the MIETEC 2.4-μm technology (standard European analog VLSI process), and for a circuit with P equaling 32 (the number of kernels), d equaling 16 (the dimension of the data space), and a precision in the memory points equal to 8 bits.

Analog memory points. We justify the use of analog memory points to store the locations of the centroids as follows. If we consider that P centroids must be stored, and that each of them has d coordinates, the chip must contain $P \times d$ memories. If each of these memories must have, for example, a precision of 8 bits as assumed in the following, storing all values digitally would lead to a large silicon area, even if we use dynamic memories in a standard CMOS technology. Furthermore, if the centroids' locations were digitally stored, the values would have to be converted locally into analog ones before using them in analog computation cells.

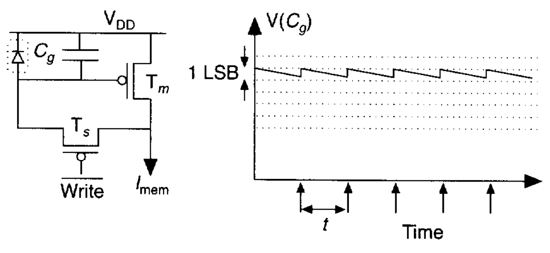


Figure 3. Analog memory point.

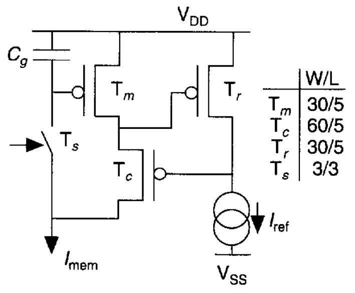


Figure 4. Regulated cascode analog memory point.

The principle of our analog memory point is to store a current as illustrated in Figure 3. When switch transistor T_s is on, the drain and gate of memory transistor T_m connect, and its gate voltage adjusts to let the input current I_{mem} flow through the transistor. When transistor T_s is switched off, capacitor C_g memorizes the gate voltage of T_m to keep the same current I_{mem} flowing through the transistor.

However, several problems occur with the cell in Figure 3. First, when transistor T_s is off, a leakage current corresponding to the source blocked junction of this transistor flows between capacitor C_g and the substrate. This current, represented by the dotted junction in Figure 3, decreases the stored voltage on C_g (referred to as V_{DD}). Second, when transistor T_s is switched off, some charges are injected in C_g, modifying its voltage as well. Finally, if the drain voltage of transistor T_m changes between the storage of current (T_s on) and its reading (T_s off), the current value also changes.

To compensate for the effects of current leakage in the blocked junction, a refreshment system sequentially reads all analog values stored on the chip and refreshes them. Both the charge injection (when switching off transistor T_s) and the leakage current in the blocked junction decrease the voltage $V(C_g)$ between V_{DD} and the gate of T_m from less than one least significant bit in refreshment period T. This LSB is measured over the whole dynamics of stored voltages on C_g. In Figure 3, both the leakage current and the charge injection have the same sign. The blocked junction injects positive charges from V_{DD} to C_g just as does the switching of T_s. We then know the sign of the slope of $V(C_g)$, as illustrated in the second part of Figure 3.

Suppose the refreshment system now reads the analog value in a memory point at T regular intervals and converts it into the smallest digital value greater than the analog one. Then the system can refresh the memory point to its initial level as illustrated, keeping the stored value fixed to a precision of 1 LSB. The same system, an analog-to-digital converter followed by a digital-to-analog one, can refresh all memory points of the circuit. It can do so provided that period T between two refreshments of the same memory point is small enough to ensure a decay in $V(C_g)$ of less than 1 LSB.

To solve the dependency problem of the current in T_m with its drain voltage, we designed the cell on the chip as a regulated cascode;[5] see Figure 4. Fixing the current I_{ref} flowing into the regulation transistor T_r maintains the drain voltage of T_m through the cascode transistor T_c. The drain voltage of T_c at the input of the cell may now vary with almost no effect on the drain voltage of T_m, because of the high gain of the loop formed by T_r and T_c. The impedance of the cell is thus high and the memorized current I_{mem} fixed.

In Figure 4, T_m operates in its linear region, reducing its transconductance g_m as well as the current variation due to the charge injection on C_g. To keep the drain voltage of T_m as fixed as possible, we must increase the gain of the loop formed by T_c and T_r. Both transistors remain in saturation, and transistor T_r operates in weak inversion (through a very small I_{ref} current) to maximize its gain (transconductance over output conductance).

The capacitance of C_g must be around 1 pF to reach an 8-bit accuracy in the stored current. To obtain this value, we added a supplementary capacitor realized between the two polysilicon layers of the MIETEC 2.4-μm technology in parallel to the gate capacitor of T_m. This gives us a constant capacitor. We set the maximum current memorized in the cell to 128 μA with one LSB corresponding to 500 nA.

We derive the approximate time constant for a current memorization in the cell by $T = C_g / g_{mm}$, where g_{mm} is the transconductance of T_m. Since g_{mm} is approximately 100 μA/V, the time constant T is about 10 ns.

We obtain the first-order output impedance of the cell by

$$Z = (g_{mr}\, g_{mc}) \,/\, (g_{dm}\, g_{dr}\, g_{dc})$$

Here g_{mr} and g_{mc} are the transconductances of T_r and T_c, and g_{dm}, g_{dr}, and g_{dc} are the output conductances of T_m, T_r, and T_c. This output impedance Z is approximately 400 Mohms in our cell.

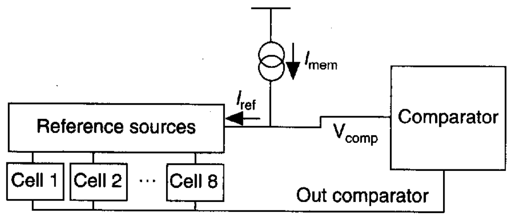


Figure 5. Refreshment system for analog memories.

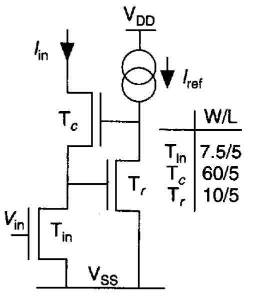


Figure 6. Regulated cascode input circuitry.

Refreshment system. As mentioned earlier, the principle of the refreshment system for analog memory points is to read the analog current stored in a cell, and to refresh it to the next upper reference current in the digital range, as illustrated in Figure 3. (The next upper voltage on C_g refers to V_{DD} corresponding to the next upper current I_{mem}.)

Figure 5 shows the architecture of the refreshment system. Cells 1 to 8 contain current sources in the power of 2, namely 1 LSB, 2 LSBs, 4 LSBs, ..., 128 LSBs. Matching these current sources is critical, since an error of 1 LSB may cause nonmonotonicity in the converter. We realize the sources by connecting elementary current sources of 1 LSB, through a design that interleaves these elementary sources to minimize the influencing oxide and technological parameter gradients in the chip.

When current I_{mem} must be measured, the comparator successively switches on cells 8 to 1, according to the successive approximation scheme. (This scheme compares I_{mem} with a reference made from 128 elementary sources. If I_{mem} is greater than the reference, the comparator compares I_{mem} with 128 + 64 elementary sources, and the most significant bit equals 1. Otherwise it compares I_{mem} with 64 elementary sources, and the MSB equals 0.)

After eight comparisons, the comparator obtains the 8-bit digital value, together with a I_{ref} current, which refreshes the memory point. At each iteration of the successive approximation scheme, V_{comp} on a high-impedance node goes high or low, depending on the sign of the subtraction of I_{ref} from I_{mem}. It then fixes the next bit in the digital value. Macq and Jespers[6] describes the converter and comparator cell in detail.

Synapse and input circuitry. The circuit in Figure 6 repeats $P \times d$ times on the chip and connects to the $P \times d$ analog memory points described earlier. The purpose of the circuit in this figure is twofold. It acts as an input of the corresponding memory point when a current must be stored. In this mode, an external input voltage V_{in} generates a current I_{in}, which is the current I_{mem} in Figure 4. Write transistor T_s switches on, and the cell stores the current in memory.

In the second operation mode, suppose I_{mem} is in the cell's memory as in Figure 4, and that it must be subtracted from current I_{in}. In the next section we show the use of this difference to compute the distance between an input vector x_i and a centroid p_j . In this mode however, the difference between currents I_{mem} and I_{in} may be allowed to flow out of these cells.

The principle of the cascode cell in Figure 6 is similar to the principle of the memory point. Cascode transistor T_c coupled with regulation transistor T_r maintains the fixed drain voltage of T_{in}. This makes I_{in} independent from the drain voltage of T_c. Moreover, since the transistor T_{in} operates in its linear region, the conversion between V_{in} and I_{in} is linear. Finally, using the same transistors first to write a value in a memory point and then compare this value to another one compensates for deviations on the absolute value of the stored currents, which may depend on the transistors' characteristics.

The transistors' operation modes in this cell are identical to the regulated cascode ones in the analog memory point; Figure 6 gives the sizes of the transistors.

Distance computation. One of the main operations that must be realized on chip is the distance computation between a d-dimensional input vector and P d-dimensional centroids. The type of distance metric used for this computation obviously influences the behavior of the algorithm; however, there is no a priori reason to prefer one type of distance metric to another. In a real-world database of learning vectors, only the shape of the regions associated with each class in the data space may influence the choice between several types of distances. Since these shapes are not a priori known in most real classification problems, we cannot foresee which type of distance will lead to the best performances. This motivates the choice for the distance metric easiest to implement in practice, that is, the Manhattan distance.

We obtain the Manhattan distance between an input vector x and a centroid p_i, $1 \le i \le P$ by

$$\text{dist}_{\text{Man}}(x, p_i) = \sum_{j=1}^{d} \left| x_j - p_{ij} \right| \tag{5}$$

Three operations must be realized: subtraction between x and p_j coordinate by coordinate, absolute value, and sum of these results over all coordinates. When a memory point is in read mode, the difference between the memorized currents $I_{mem j}$ and the input currents $I_{in j}$ in Figure 7 ($1 \le j \le d$) flows out

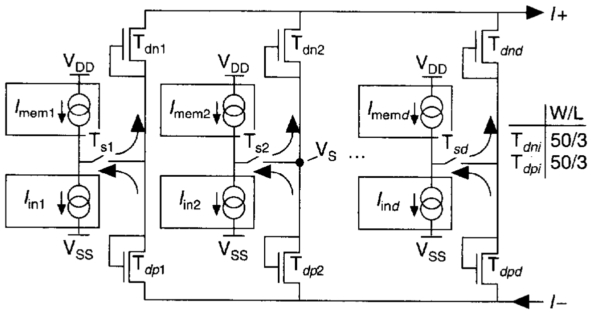


Figure 7. Computation of the Manhattan distance between the input vector and one centroid.

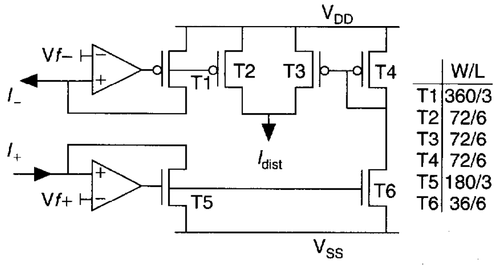


Figure 8. Absolute value of the sums of currents.

from the cells (transistors T_{sj} are switched on). This current may be positive or negative. Depending on its sign, it is directed to one of the two summation current lines in Figure 7. Finally, we must compute the difference between these two sums to complete the implementation of Equation 5.

The circuit illustrated in Figure 8 implements this current difference, and sets the respective voltages of current lines I_+ and I_- (in Figures 7 and 8) to fixed values V_{f+} and V_{f-}. This lets us keep the output of each cell in Figure 7 in a limited voltage range to ensure the functionality of the memory points and the input circuits. We fixed V_{f+} and V_{f-} to about 1.6V and 3.4V to let the currents flow through the diodes T_{dnj} and T_{dpj}, $1 \leq j \leq$ in Figure 7.

Both of these values are not critical, so we roughly generate them by dividing the power supply voltage. The influence of technological parameters and temperature is negligible, provided that the voltages remain constant during the use of the chip. We determined the difference between these two values to avoid a constant current flowing through the diodes T_{dnj} and T_{dpj}, by keeping this difference smaller than the sum of the absolute values of the two threshold voltages.

The current mirrors in Figure 8 sum the absolute values of I_+ and I_- to provide I_{dist}, the distance between the input vector and the memorized centroid. The dynamics of currents I_+ and I_- may be important. Since the maximum stored current in each memory point is 256 × 500 nA = 128 μA, their sum may be as high as 16 × 128 μA ≅ 2 mA. The mirrors used in the cell of Figure 8 divide the current dynamics by 10, to get acceptable currents for the winner-take-all and kernel cells.

Figure 9 represents the simulation of one distance computation cell connected to the circuit of Figure 8. We assume a memorized current of 75 μA, while V_{in} of the input cell ranges from 2V to 4.5V. The first curve (Figure 9a) represents the current in the input cell, which ranges from 20 to 180 μA (the expected dynamics). The second curve (Figure 9b) represents the output voltage V_s of the cell in Figure 7 (between transistor T_s and the diode-connected transistors). The last curve (Figure 9c) represents the output current after inversion by the circuit of Figure 8.

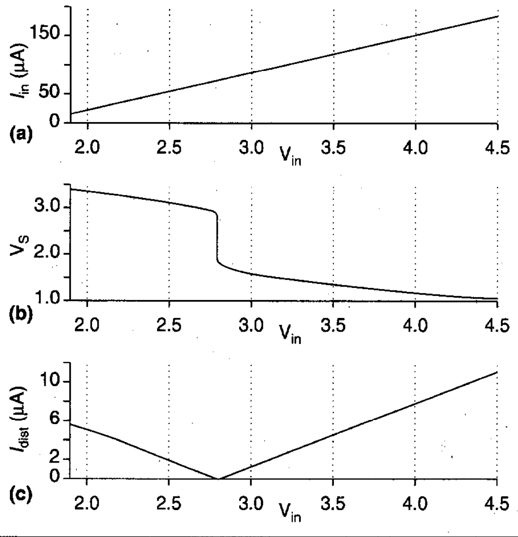


Figure 9. Simulation of Manhattan distrance computation.

Winner-take-all. Lazzaro et al.[7] first proposed the principle of a winner-take-all circuit with current inputs, as shown in Figure 10, next page. V_A is the gate voltage of all transistors T_{1i} and the source voltage of all T_{2i}, $1 \leq i \leq P$. Since all transistors T_{1i} have the same gate voltage, their drain voltage will adjust to let the $I_{\text{in}i}$ currents flow through the transistors. Since the currents are different (the maximum must be chosen), only one T_{1i} is in saturation, while the others are in their linear region. T_{2i} has a gate voltage lower than the others, and catches the main part of the current I_{ref} since the source voltages of T_{2i} are fixed to V_A too. The output voltage of each cell detects the winner.

Lazzaro et al. only uses transistors in weak inversion. In our circuit however, because of the dynamics of the input currents, and also to reduce the time constants, we prefer transistors in strong inversion. We added a Schmitt trigger at the output of each cell, to avoid oscillations between winners should two currents be approximately identical. We selected the threshold of these Schmitt triggers to ensure at most one output would be high, with the possibility that all outputs would remain low if several cells share current I_{ref} and if two or more input currents are similar.

The sizes of the transistors given in Figure 10 are valid for input currents from 0 to 200 μA, which is the range of the currents coming from the distance computation cells. We chose an I_{ref} current of 1 μA. However, the LVQ algorithm necessitates the selection of the centroid whose distance is the minimum from the input vector, and the winner-take-all block of Figure 10 selects the maximum of its input currents. Therefore, we need to subtract current I_{dist} of Figure 8 from a fixed value to get I_{ini} in Figure 10; in our circuit I_{ini} equals 200 μA − I_{disti}. The second winner-take-all, which selects the largest estimate of probability densities from among classes for the KBC algorithm, does not need this subtraction. With the sizes of transistors given in Figure 10, we can discriminate between two currents with 1 percent difference, whatever the absolute value of these currents (0 to 200 μA).

Kernel functions. Recent developments in the theory of KBC algorithms[8] show that we can greatly improve the quality of probability density estimations by adjusting two kinds of parameters in the Gaussian kernels. The first one is classically its width factor. A second one, which must be adjusted depending on the dimension of the data space, determines the tail curvature of the Gaussian function, that is, the rate at which the kernel function drops off.

Figure 11 shows the differential pair used to realize a Gaussian-like kernel. Note that the exact kernel shape is not critical for the approximation of probability densities as soon as two such parameters can be adjusted. Moreover, only half of the Gaussian function must be realized, since its argument is always positive (distances). We will thus use the nonlinear characteristics of a differential pair to evaluate the Gaussian-like functions.

In Figure 11, flowing the argument of the kernel function—namely the current I_{dist} in Figure 8—into a transistor in its linear region generates V_{in}. V_{ref} determines the width of the kernel, while modifying V_c, which acts on the conductance of T_3, adjusts its curvature. In the implementation, three transistors connected in parallel with W/L sizes of 3/160, 3/80, and 3/40 form T_3. Only logic voltages (0 and 5V) are allowed on their gates to modify the shape of the kernels, so that the values may be memorized in static memory points. We keep on at least one of these three transistors.

Another solution for T_3 would be to implement only one transistor and to control its conductance by varying its gate voltage. However, this transistor must work in a resistorlike linear region to ensure good behavior of the cell. This prohibits the use of low voltages on the gate of T_3, which seriously limits the dynamics of the slope in Figure 13. Figures 12 and 13 show a simulation of the kernel function for a T_3 of size 3/160 with 5V on its gate and V_{ref} between 0.5V and 2.5V. The figures also show different combinations of switching on or off transistors T_3 for V_{ref} fixed to 1.5V. Current I_0 stays fixed to 1 μA.

Cascadability of the chips. The fabrication yield of analog integrated circuits limits the size of our chip. As mentioned earlier, we calculated all sizes for a number P of

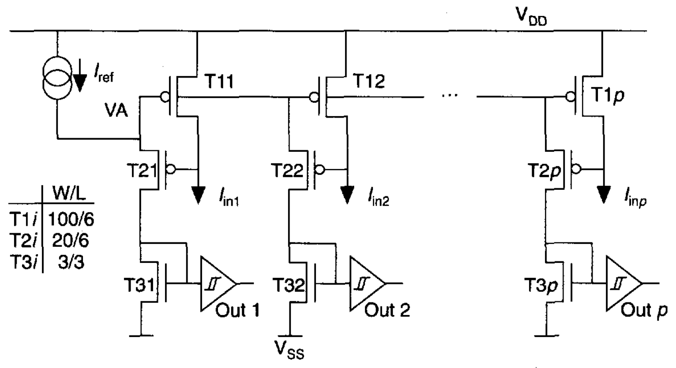


Figure 10. Winner-take-all circuit.

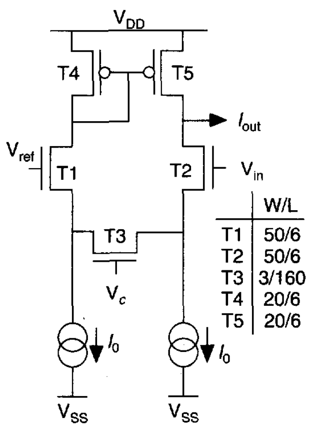


Figure 11. Kernel Gaussian-like function.

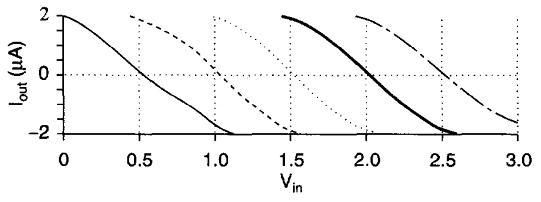


Figure 12. Kernel simulation for T_3 fixed and V_{ref} between 0.5V and 2.5V.

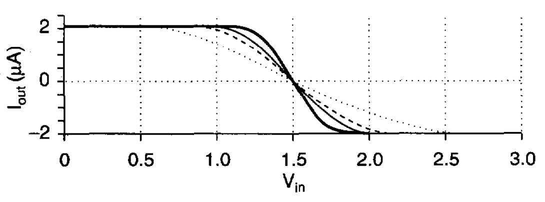


Figure 13. Kernel simulation for V_{ref} fixed to 1.5V and different T_3's switched on.

kernels equal to 32, and for a space dimension *d* equal to 16. While several chips cannot be directly cascaded to increase *d*, it is possible for the number of *P*. All operations in the chip are separated from one kernel to another, except the two winner-take-all blocks. When these blocks are connected together between chips, a set of several circuits can implement the LVQ and KBC algorithms with an increased number of kernels.

The first winner-take-all block selects the winner from among all distances computed in a chip. To select a winner among distances computed in two different chips, a supplementary input at the winner-take-all block in the second chip enters the distance already selected as the winner in the first chip. Depending on the selected winner in the second chip, the digital control section determines whether the winner is in the first or the second chip, and thus which is the winning centroid. In reality, two supplementary inputs at the first winner-take-all are provided in each chip, for possibile connection in a tree structure, to minimize propagation delays between chips. To avoid the influence of technological parameter mismatches between two different chips on the current flowing into the winner-take-all blocks, we must avoid voltage-to-current conversion from the input of the chip. In this case, a supplementary input cell identical to the regulated cascode input circuitry of Figure 6, but diode-connected, must be used at the inputs of the analog processor, to allow current as well as voltage input.

The problem of the second winner-take-all in Figure 2 is different. Its inputs are the current lines by class, which do not have to be duplicated if two chips are connected. Rather, we must sum the currents flowing on the corresponding lines in the two chips, the winner-take-all having then to select the winner among these sums. We simply connect the input currents of the two winner-take-alls, as the gates of T_{1i} (voltage V_A). In this way, we connect T_{1i} in the two chips in parallel and drive twice the current in a unique chip. Technological parameter mismatches between different chips do not directly influence the selection of the winner when connecting several winner-take-alls in parallel. We can measure digital outputs after the Schmitt triggers in any of the two chips. Obviously, we can apply the same principle to more than two chips. Finally, cascading chips will increase the response times of the winner-take-alls, by adding capacitances due to pads, wires, and packaging.

Technology limitations

When designing a complex analog VLSI circuit, we must carefully consider the accuracy and matching of components, as well as the parasitic effects (leakage currents, charge injection) in some cells. We encountered two kinds of limitations during conception of the chip. The first limitation concerns the sizing of the regulated cascode analog memory points; the second concerned algorithmic considerations of the accuracy in the different parts of the chip.

Sizing regulated cascode cells. The simple analog memory point, illustrated in Figure 3, presents three main drawbacks: blocked junction leakage currents, channel length modulation, and charge injection and clock feedthrough when switching off T_s.

We propose reducing these parasitic effects with

- periodic refreshment to overcome the junction leakage effects,
- cascode-like design to increase the output impedance, and
- a large storage capacitor coupled with a small transconductance for T_m, to decrease the charge injection and clock feedthrough effects.

Others have proposed some tricky solutions to reduce the last parasitic effect,[9-11] but for simplicity we did not implement them in our chip. Our refreshment system requires, however, the combined leakage currents and charge injection effects (eventually compensated) to be smaller than 1 LSB and of known sign.

The block used for the analog memory point is in fact a regulated cascode, as illustrated in Figure 14 (next page). We multiply the output impedance of T_m by the gain loop formed by T_c, the cascode transistor, and T_r, which regulates the loop. To obtain high gain, both transistors must have large transconductances (large W/L) and high output imped-

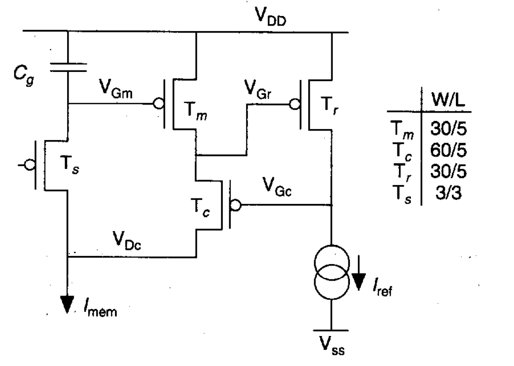


Figure 14. Regulated cascode analog memory point with voltage conventions.

ances: They work in saturation. With a regulated cascode cell, the gain loop is so big that we can afford to put T_m in a linear region to decrease its transconductance and thus the charge injection and clock feedthrough effects.

The charge injection and the clock feedthrough occur when T_s switches off. When the switch is on, the channel of this transistor is full of carriers, and the last ones must escape to the drain and the source when the transistor is turning off. This phenomenon is known as charge injection. The clock feedthrough effect results when storage capacitor C_g and the gate-to-source overlap capacitor of T_m form a parasitic capacitor divider. A part of the clock signal is injected on C_g through this path. We roughly compute the total amount of charges produced by these two phenomena by

$$Q \approx (WLC_{ox}/2)(V_G - V_{T0} - nV_s) + WC_{ov}\Delta V_{clk}$$

where ΔV_{clk} is the drop voltage on the gate of T_s. In this equation, the first term in the right member represents the charge injection and the second one, the clock feedthrough. All parameters concern T_s; C_{ox} is the gate capacitor per unit area, and C_{ov} is the gate-to-source overlap capacitor per unit length.

This charge Q produces a voltage drop on the storage capacitor

$$\Delta V_{Gm} = (Q / C_g) \qquad (6)$$

If C_g is the gate capacitor of the memory transistor T_m, the voltage drop strongly depends on its DC mode. From deep linear regime to strong saturation, the gate-to-source capacitor evolves from $WLC_{ox}/2$ to $3WLC_{ox}/2$, and the gate-to-drain capacitor from $WLC_{ox}/2$ to zero.[12] To avoid a variable capacitor in case of linear regime or a nonefficient use of the silicon area in case of saturation, we chose a double polysilicon capacitor, available in MIETEC 2.4-μm technology.

The voltage drop ΔV_{Gm} on the storage capacitor and, of course, on the gate of T_m modify the stored current. This little current drop due to the switching is

$$\Delta I = g_m \Delta V_{Gm}$$

where I is the drain current in T_m. The transconductance g_m of T_m also depends on its DC mode:

$$g_m = \sqrt{\frac{2I\beta_m}{n}} \quad \text{if } T_m \text{ is saturated}$$
$$g_m = \beta_m V_{DS} \quad \text{if } T_m \text{ is linear}$$

where β_m is the conductance parameter of T_m and n is the substrate effect parameter. A memory transistor in linear mode seems better than a saturated one with the injection and clock feedthrough problems for two reasons. The transconductance is constant in linear regime and thus also the variation of the current with the hypothesis of a constant charge injection from the switch. This transconductance is smaller than in saturation.

Of course, if T_m is in linear mode, each small modification on the drain voltage of this transistor will drastically modify the stored current. The gain loop formed by T_c and T_r must then be very high. This solution is possible if we use saturated transistors near to or in weak inversion.

Sizing the transistors. The regulated cascode analog memory can be in write mode, with the gate of T_m connected to the drain of T_c via T_s, or in read mode when this connection is open. The circuit must be designed such as T_c does not leave saturation and T_m does not turn into saturation.

In writing mode, the equation and the associated condition of the drain current of T_m are

$$I = \beta_m V_{Gr}[V_{Gm} - V_{T0} - (n/2)V_{Gr}] \qquad (7)$$

$$V_{Gr} \leq [(V_{Gm} - V_{T0})/n] \qquad (8)$$

and those of T_c are

$$I_D = (\beta_c/2n)(V_{Gc} - V_{T0} - nV_{Gr})^2;\ V_{Gm} \geq [(V_{Gc} - V_{T0})/n]$$

Here voltage notations are indicated in Figure 14 (all voltages being referred to V_{DD}), β_c is the conductance parameter of T_c, and V_{T0} is the threshold voltage of P-type transistors.

The cell always needs a bias current to work properly. Increasing V_{Gm} provides a greater ratio between the useful current and the bias one, but this gate voltage cannot be too high to avoid some tricky problems on the input circuitry and on T_c. Equations 7 and 8 give the maximum value for β_m, mainly depending on the gate voltage V_{Gr} and on the maximum drain current. On the other hand, the maximum and the minimum of V_{Gm} determine the maximum and minimum gate voltages of T_c and the minimum of its conductance parameter β_c. We also have Equation 6 for the injected current produced by the switching of T_s. If we take all these relations and minimize the surface of T_m, T_c and C_g, considering a minimum drawing size of 5 μm, we can compute all the values only in terms of V_{Gm} and I_{LSB}, the current of 1 LSB in the memory point (fixed to 500 nA).

The value of V_{Gr} is more difficult to optimize accurately. A higher V_{Gr} means higher gate voltages on T_m and T_c, but decreases on both conductance parameters β_m and β_c. We fixed the value of V_{Gr} to $2V_{T0}/3$ for our circuit. An optimization process taking all these equations and limits into consideration gave a bias current of about 20 μA, a storage capacitance of 1 pF, and transistor sizes near the values shown in Figure 4.

In read mode, the gate of T_m is disconnected from the rest of the circuit. The conditions are similar: T_m and T_c are in linear and saturated modes; but there is no direct relation between both transistors. The more restricting condition is the drain voltage of T_c, which cannot be too low (referred to as V_{DD}), to avoid putting the transistor in linear mode and drastically dropping the impedance of the cell.

The input circuitry is a regulated cascode cell in read mode, made of N-type transistors. The drain voltage of this cascode cell cannot be too high (referred to as V_{SS}), to avoid putting the cascode transistor of the memory point in linear region. It cannot be too low anymore, to maintain the input circuitry in saturation. The input transistor is in linear mode to take advantage of the linear voltage-to-current conversion characteristic, since voltage sources directly drive the corresponding inputs of the circuit.

Accuracy in the different chip cells. The accuracy needed for computations in a chip is a key problem for the analog designer. The same accuracy is not necessary in the different parts of our chip, if we consider the requirements of the problem to solve itself rather than directly focusing on the different cells of the chip. We had decided earlier to design the analog memory points to have 256 different levels, that is, a precision of 8 bits. To obtain this accuracy in all the circuit cells is, however, quite impossible and unnecessary as well.

A precision of 8 bits in the memory points (and in the input circuitry) means that 256 different values are possible for each coordinate of a centroid, that is, 256^d different possible locations. Consider the distance computation between a centroid and an input vector. When adding the different components in the Manhattan distance computation (Equation 5), we sum the absolute errors (1 LSB per synapse). However, we must not forget that this is an absolute error, which does not depend on the memorized value itself. The relative error on one memorized current is thus greater than 1/256, just as the relative error on the sum. This first consideration already justifies the idea that a precision of 8 bits in further computations, and especially in the current mirrors of Figure 8, is unnecessary.

On the other hand, the question may arise about a reduced precision in the distance computation that does not have a negative influence on the performances of the algorithm. If the current mirrors in Figure 8 have a relative precision of t bits (t is around 6 in our implementation), one of the two currents I_+ and I_- may be corrupted up to a factor $1/2^t$ for the other current in the computation of the Manhattan distance. In a classification problem however, only the smallest distances between an input vector and the centroids will further influence the computations. In kernel classification, the kernel evaluation of large distances gives negligible results.

In LVQ algorithms, where the first winner-take-all function must be used, a nonnegligible error on the input current of the winner-take-all could be generated. We find the difference between 1) the current representing the distance between the input vector and a centroid, and 2) a fixed value (to use a winner-take-all instead of a looser-take-all). However, this error, measured in terms of a fixed-error current, only produces a sensible influence when the input current at the winner-take-all is small. It is small when the input vector is far from a centroid, and thus nonrepresentative.

In both cases, the absolute error made on the distance computation will thus sensibly influence the input current (compared to the influence of the restricted precision in the memorized currents) only when the smallest distances are themselves large. For example, suppose t equals 6, and the resulting current itself is four times the whole range of one memorized current. Then, the absolute error made on the resulting current in the distance computation would equal the error made in one memory point. This is obviously seldom the case, especially in a classification system in which the largest accuracy must be found in the regions where distribution of different classes overlap, that is, where the distances between an input vector and a centroid are small. Furthermore, one can imagine that, even if it should happen that the smallest computed distances are large compared to the dynamics of one memorized current, this would mean that the input vector to classify would be far away from all centroids memorized in the network. Thus an accurate decision in its classification does not make sense.

All these considerations justify the limited precision in the current mirrors of Figure 8, which certainly need not be designed to obtain the same 8-bit precision as in the analog memory points. We assumed a 6-bit precision in the distance computations, the winner-take-all blocks, and the kernels in our chip.

AS CLASSIFICATION PROBLEMS GROW in dimension, class number, and learning set size, the computation load increases in such a way that solving real-time problems on stand-alone machines becomes very difficult. Moreover, the need for portable systems that are not continuously connected to a host computer encourages the development of chip-based classification systems.

Our analog architecture realizes the operative part of a classifier, implementing LVQ-like and kernel-based classifiers. We have implemented and are testing the analog cells and plan to produce a fully parallel processor in the near future. This chip must connect to a conventional finite-state machine implementing the control section to form an efficient, stand-alone parallel classification system.

Acknowledgments

We thank the reviewers for their appreciated comments and suggestions. The ESPRIT-BRA project 6891, Elena-Nerves II, supported by the Commision of the European Communities (DG XIII), funded part of this work.

References

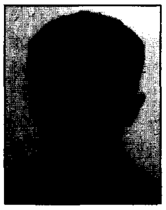
1. T. Kohonen, *Self-Organization and Associative Memory*, 3rd ed., Springer-Verlag, Berlin, Heidelberg, Germany, 1989.
2. Y. Linde, A. Buzo, and R.M. Gray, "An Algorithm for Vector Quantizer Design," *IEEE Trans. Comm.*, Vol. 28, Jan. 1980, pp. 84-95.
3. T. Cacoullos, "Estimation of a Multivariate Density," *Annals Inst. Stat. Math.*, Vol. 18, 1966, pp. 178-189.
4. P. Comon, "Classification supervisee par reseaux multicouches," *Traitement du Signal*, [Supervised Classification by Multilayer Networks], Vol. 8, No. 6, Jouve Publishers, Paris, Dec. 1991, pp. 387-407.
5. C. Toumazou, J.B. Hugues, and D.M. Pattulo, "Regulated Cascode Switched-Current Memory Cell," *Electronic Letters*, Vol. 26, No. 5, Mar. 1990, pp. 303-305.
6. D. Macq and P. Jespers, "Charge Injection in Current Copier Cells," *Electronic Letters*, Vol. 29, No. 9, Apr. 1993, pp. 780-781.
7. J. Lazzaro et al., "Winner-Take-All Networks of O(N) Complexity," *Advances in Neural Information Processing Systems*, Vol. 1, D.S. Touretzky, ed., Morgan Kaufmann Publishers, Inc., Palo Alto, Calif., 1989, pp. 703-711.
8. P. Comon, J.L. Voz, and M. Verleysen, "Estimation of Performance Bounds in Supervised Classification," *Proc. European Symp. Artificial Neural Networks*, De Facto Publications, Brussels, Apr. 1994, pp. 37-42.
9. G. Wegmann, and E.A. Vittoz, "Basic Principles of Accurate Dynamic Current Mirrors," *IEE Proc.*, Vol. 137, No. 2, Apr. 1990, pp. 95-100.
10. C. Eichenberger and W. Guggenbuhl, "On Charge Injection in Analog CMOS Switches and Dummy Switch Compensation Techniques," *IEEE Trans. Circuits and Systems*, Vol. 37, No. 2, Feb. 1990, pp. 256-264.
11. D. Vallancourt, Y. Tsividis, and S.J. Daubert, "Current-Copier Cells," *Electronic Letters*, Vol. 24, No. 25, Dec. 1988, pp. 1560-1562.
12. Y. Tsividis, *Operating and Modeling of the MOS Transistor*, McGraw-Hill, N.Y., 1987, pp. 310-328.

Michel Verleysen is a senior research assistant of the Belgian National Fund for Scientific Research (FNRS) at the Catholic University of Louvain in Belgium.

Verleysen received his engineering and PhD degrees from the Catholic University of Louvain. He has authored 30 neural network publications and organized the European Symposium on Artificial Neural Networks. He is a member of the IEEE and the International Neural Networks Society.

Philippe Thissen is studying for his PhD in neural networks on a Belgian IRSIA (Institut pour l'Encouragement de la Recherche Scientifique dans l'Industrie et l'Agriculture) fellowship in the Microelectronics Laboratory at the Catholic University of Louvain. His current interests include neural networks for classification tasks and analog implementations of these structures.

Thissen received his electrical engineering degree from the same university. He is a member of the IEEE.



Jean-Luc Voz is working under the frame of the European Community's ESPRIT Elena project in evolutive neural networks and studying toward his PhD degree in this field in the Microelectonics Laboratory at the Catholic University of Louvain. His research activities and interests include pattern recognition, neural networks, and signal processing.

Voz received his electrical engineering degree from the Catholic University of Louvain and is a member of the IEEE.

Jordi Madrenas' biography and photograph appear on p. 59 of this issue.

Reader Interest Survey

Indicate your interest in this article by circling the appropriate number on the Reader Service Card.

Low 159	Medium 160	High 161