NEURO-NIMES 91

# FROM THEORY TO SILICON : AN EFFICIENT PROCEDURE FOR THE DESIGN OF "NEURAL" CLASSIFIERS AND ITS APPLICATION TO THE AUTOMATIC RECOGNITION OF HANDWRITTEN DIGITS

**Stefan Knerr, Léon Personnaz, Gérard Dreyfus**
**Ecole Supérieure de Physique et de Chimie Industrielles**
**de la Ville de Paris**
**Laboratoire d'Electronique**
**10, rue Vauquelin**
**F - 75005 PARIS - FRANCE**

**Phone: (1) 40 79 45 41**

## ABSTRACT

We describe a procedure for simultaneously building and training a neural network classifier, and we present its application to the automatic recognition of handwritten digits from two large data bases. An application-specific integrated circuit, which is in the foundry phase, is briefly described.

**Keywords**: classification, growth algorithm, training algorithms, network structure, character recognition, zip code, hardware implementation, ASIC.

## RESUME

Nous décrivons une procédure qui permet d'efffectuer simultanément la construction et l'apprentissage d'un réseau de neurones pour la classification, et nous présentons l'application de cette procédure à la reconnaissance automatique de chiffres manuscrits provenant de deux bases de données de grande taille. Nous décrivons un circuit intégré spécifique qui est en cours de fabrication.

**Mots clés:** classification, algorithme de croissance, algorithmes d'apprentissage, structure de réseau, reconnaissance de caractères manuscrits, code postal, réalisation électronique, circuit intégré spécifique.

## 1 - INTRODUCTION

During the past few years, multilayer perceptrons have been widely advocated as classifiers; nevertheless, whether there is any fundamental reason why multilayer perceptrons should perform significantly better than classifiers of similar computational complexity (or even of lower complexity) is still a point of debate. Neural networks, however, may have an edge in ease of hardware realization or parallel software implementation. Thus, trying to discover ways of designing a neural network classifier with the best performance-to-cost ratio is a challenge of practical engineering importance. The performance of a classifier is measured in terms of its generalization ability; its cost is measured in terms of number and type of units (complexity of the non-linear transfer characteristics of the neuron), of number of trainable connections, of training time and of classification time.
Since the issue of determining explicitly a satisfactory neural network structure is crucial for the future of neural networks operating as classifiers, for anything but toy problems, several approaches have been proposed. The simplest idea is the brute force approach, whereby the number of hidden layers and the number of hidden units is varied more or less systematically, until satisfactory results are obtained; this is a wasteful and conceptually unsatisfactory procedure. Two alternative approaches emerged around 1989: the first one consists in starting with a very small network and adding neurons [1-4]; the second one consists in starting with a large network and pruning it to an "optimal" size [5].
The present procedure belongs to the first approach: it starts with one neuron per class, and builds the network in three steps which will be detailed below. Its salient features are the following:
- the network has *one single layer of trainable connections;* therefore, training is *fast;*
- the additional layers perform *explicit boolean functions;* therefore, these layers require *no training,* and they can be implemented in hardware with standard logic gates;
- the procedure *gives insight into the complexity of the problem,*
- the resulting network uses neurons *with binary outputs,* which makes hardware implementations straightforward.

The procedure was applied to a real-life problem: the automatic recognition of handwritten digits from two large data bases. We present the results which were obtained and describe briefly an application-specific integrated circuit which is being manufactured.

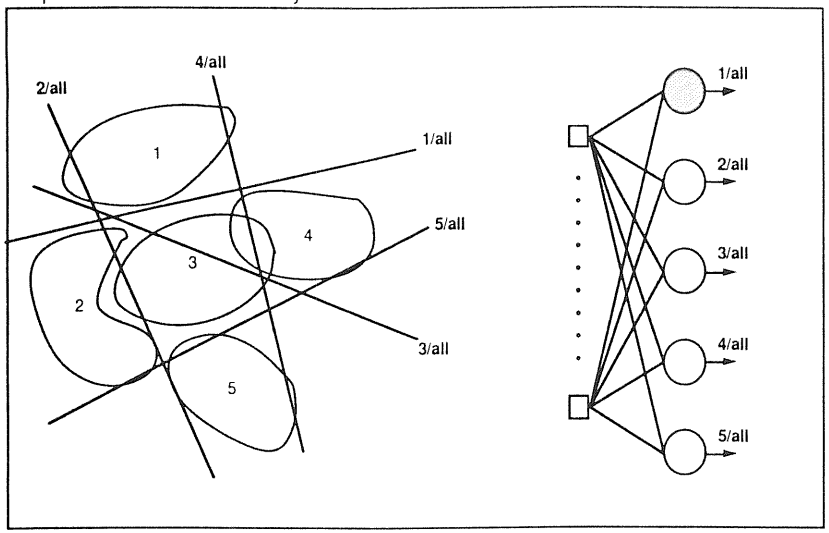
## 2 - THE BUILDING AND TRAINING PROCEDURE

This section provides a short description of the building and training procedure; more details are to be found in [1]. It is basically a "divide and conquer" procedure, based on two simple ideas:
(i) when dealing with a multiclass problem - say, ten classes for ten handwritten or spoken digits - the outputs of the network are "grandmother cells"; in other words, the network has to separate each class from all others. This is usually difficult in real-life problems, where each class is not necessarily linearly separable from all others. Conversely, it might well be that classes are *pairwise* linearly separable, in which case a set of *independent* linear separators would solve the problem;
(ii) if the classes are not pairwise linearly separable, each pair of classes may be separated using a recursive partitioning procedure, in the spirit of decision trees [6].

### 2.1 - General description

The building and training phase proceeds in three steps, illustrated on Figures 1a and 1b:
- in a first step, linear separation of each class from all others is attempted; this results in successful neurons (such as neuron denoted by "1/all" on the upper diagram of Figure 1a), which are kept, and in unsuccessful neurons (such as the other neurons on the upper diagram of Figure 1a), which are discarded;
- in the second step, *pairwise linear separation* of the classes which were not separated during the previous step is attempted; again, the resulting successful

**FIRST STEP: LINEAR SEPARATION OF EACH CLASS FROM ALL OTHERS**
Training until validation.
Keep successful neurons only.




**SECOND STEP: NEURONS FOR PAIRWISE SEPARATION**
Linear separation of pairs of classes.
Training until validation.
Keep successful neurons only.

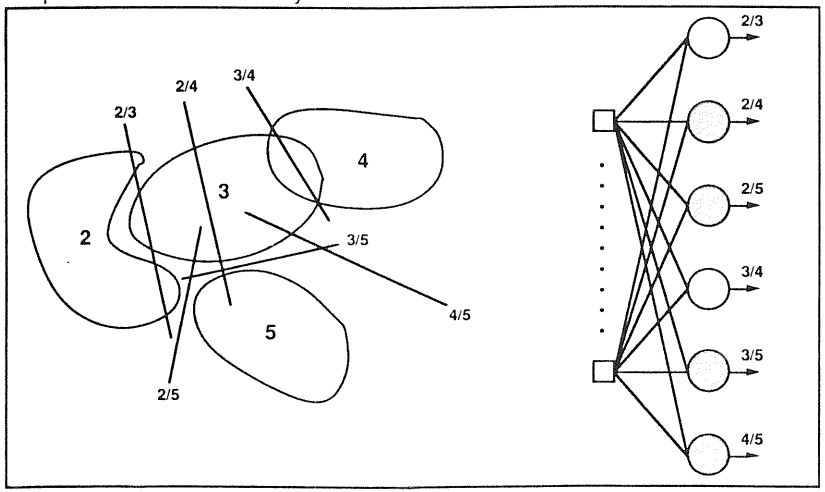
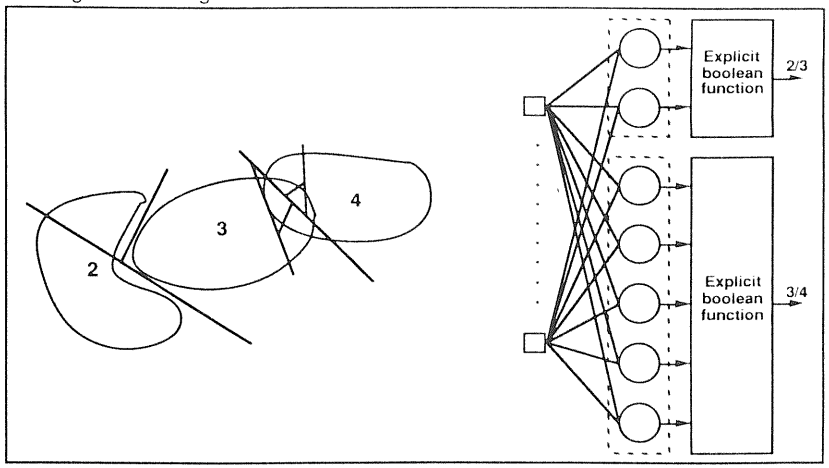



Figure 1a

**THIRD STEP: SUBNETWORKS FOR PAIRWISE SEPARATION**
Piecewise linear decision boundaries.
Building and training until validation.




**FINAL NETWORK:**

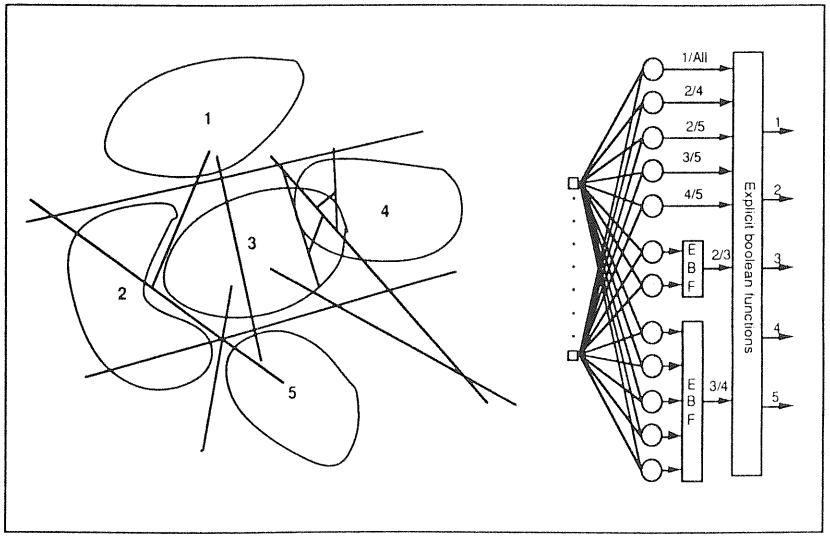



Figure 1b

neurons (such as neurons denoted by "2/4", "2/5", "3/5", "4/5" on the bottom diagram of Figure 1a) are kept, whereas the unsuccessful ones are discarded;
- in the third step, the classes which are still not separated (either because they are non-linearly separable, or because they are overlapping), such as classes 2, 3 and 4 on Figure 1b, are separated pairwise by *piecewise linear* decision surfaces. This is performed by building a binary decision tree through successive splits of the input space until all examples of the training set are correctly classified, in the spirit of [6]. The third step might also conceivably be performed by a set of independent multilayer perceptrons trained by backpropagation *on pairs of classes only.*
- the final decision is made by ANDing the outputs of the neurons created during steps 1 and 2, and the outputs of the boolean functions generated by step 3.

### 2.2 - Training algorithms

As mentioned above, the three steps require *single-layer training only.* Steps (1) and (2) make use of the generalized delta rule [1]. The modification of the coefficient $c_i$ of a neuron when example $(\mathbf{x}^k, d^k)$ from the *training set* is presented, is expressed by:

$$\Delta c_i(k) = \frac{\mu}{||\mathbf{x}^k||^2} (d^k - s^k)\, f'(v^k)\, x_i^k$$

with $v^k = \sum_{j=1}^{n} c_j(k-1)\, x_j^k$, $s^k = f(v^k)$ being the corresponding output and f(.) being the sigmoidal non-linearity.

Even when training a single neuron, there is a danger of "overspecialization": the final position of the separating hyperplane may be biased by marginal examples. Therefore, performance estimations on a *validation set* should be carried out while the neuron is trained. The best solution, with respect to the performance estimation on the validation set encountered during training, is kept.

Step (3) attempts to optimize an information-entropy-like measure. More details are to be found in [7].

In the first two steps, training requires sigmoidal non-linearities f(.), but, once training is performed, the classifier makes use of neurons with *binary* outputs for the subsequent boolean decisions. This greatly facilitates hardware implementations. It should be noticed that, in a multilayer perceptron with hidden neurons, the replacement of the sigmoidal neurons by binary neurons is not possible: the hidden units do not operate as a piecewise linear separator, but only contribute to the nonlinear function which is implemented by the *complete* multilayer perceptron.

After training, the performance of the classifier is estimated on a *test set,* independent of the training set and of the validation set.

## 3 - APPLICATION TO THE RECOGNITION OF HANDWRITTEN DIGITS

The above procedure was applied to the automatic recognition of handwritten digits [7]. Two data bases were used in our investigations: a "European " data base featuring 8,700 presegmented handwritten digits, and the US Postal Office Data Base, featuring 9,000 digits. Some examples from the European data base are shown on Figure 2. The left columns show the original digits, the right columns show the digits after the normalization procedure described in the next section.

### 3.1 - Preprocessing

In the case of the European data base, the digits were presegmented. The zip codes of the US Postal Office data base were segmented by a semi-automatic procedure.

For classification, two data representations were used:

(i) a pixel representation was obtained by simple normalization of the segmented data to 16 by 16 pixels with 16 grey levels; the effect of the quasi-linear normalization is shown on Figure 2; (ii) a feature representation obtained by skeletonization and feature extraction (line segments) using masks, followed by normalization; the resulting representation was a vector of 256 components encoded on 4 bits. Thus, the data input to the classifier was in the same format, whichever representation was used.

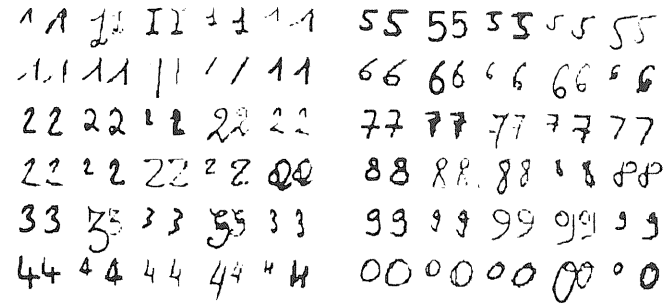


Figure 2

### 3.2 - Classification

3.2.1 - Training of the classifier

After preprocessing, the data bases were randomly divided into training set, validation set and test set. We used a larger training set for the US Postal Office than for the European data base, since the former exhibits more variations in writing styles than the latter. 80% of the US Postal Office data base was used for training and validation, while 50% of the European data base only was found necessary for a satisfactory sampling of the writing styles.

3.2.2 - Structure of the classifier resulting from training

The automatic building and training procedure was applied to both data bases and both data representations. Interestingly, in all four cases, the procedure stopped after the second step, thereby indicating that the classes of the training set are pairwise linearly separable. The resulting network architecture is very simple (Figure 3): a single layer of 45 neurons is fully connected to 257 inputs; the final decision is made by ten AND gates. The classifier has 11,565 trainable coefficients. This figure might lead to the idea that the number of coefficients is larger than the number of examples. This is not correct: it should be remembered that *each neuron is trained separately* on two classes; thus, each neuron has 257 coefficients which are trained with 1,400 examples in the case of the U.S. Postal Office data base and with 870 examples in the case of the European data base, which are reasonable ratios for a meaningful positioning of each hyperplane; note that the number of examples used for training is limited by the present size of the data base, not by the computation time, which is inherently much shorter than that of backpropagation.

It is clear that, in order to assess the performance of a classifier, three figures of merit must be considered: the number of well classified items, the number of errors (misclassified items), and the number of rejected items. For many applications, it is more important to minimize the number of errors, than to maximize the number of well classified items, the price being a higher rejection rate. Therefore, a realistic recognizer should implement a flexible rejection mechanism. In order to achieve this, the values of the potentials are taken into account for the final decision made by the AND gates: a small value of a potential indicates an ambiguous situation.

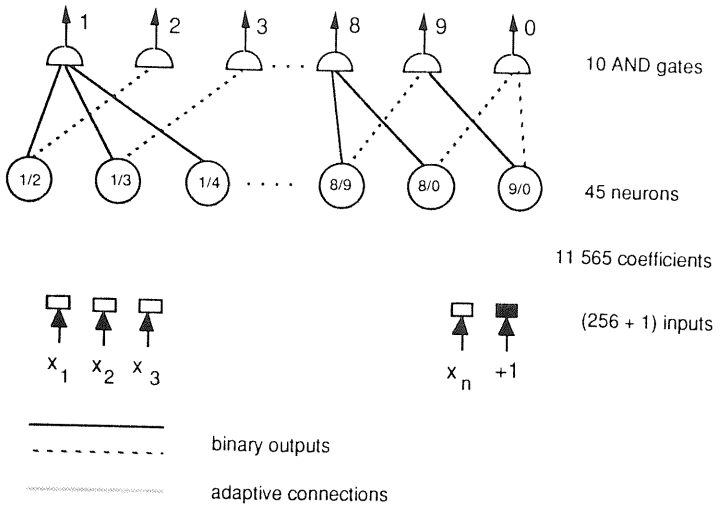



Figure 3: Network architecture

Figure 4 illustrates neuron (i/j), separating class i from class j and the rejection mechanism: each neuron compares its potential $v_{(i/j)}$ to a common threshold $\theta$; the two binary outputs $s_{(i/j)j}$ and $s_{(i/j)i}$ are determined by the following rules:

- if $v_{(i/j)} < -\theta$ , then $s_{(i/j)j}=1$ and $s_{(i/j)i}=0$ ;
- if $v_{(i/j)} > \theta$, then $s_{(i/j)j}=0$ and $s_{(i/j)i}=1$ ;
- if $|v_{(i/j)}| < \theta$ then $s_{(i/j)j}=s_{(i/j)i}=0$; indicating an ambiguous input pattern.

The final decision of the network is made as follows:

- if $s_{(i/j)i}=1$ for all j, the output of the AND gate "i" is one, and the input pattern is assigned to class i ;
- if all AND gates have zero outputs, the input pattern is rejected.

A low threshold results in a high percentage of well classified examples, whereas a high threshold yields a low error rate.

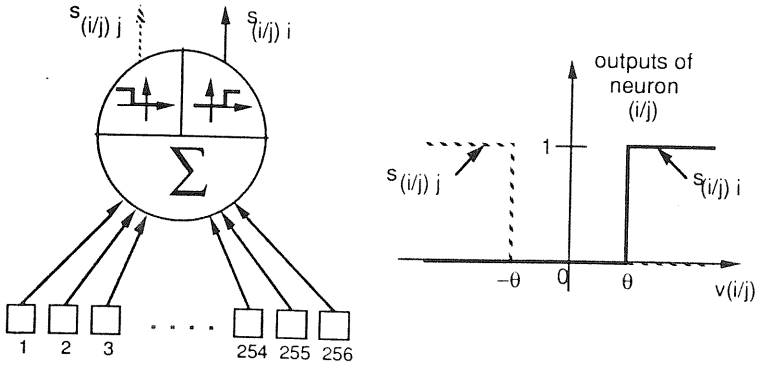



Figure 4: Neuron (i/j) and rejection mechanism.

3.2.3 - Classification results

Tables 1 and 2 show the results obtained on the European data base and the U.S. Postal Office data base respectively. All results are averages over 5 different partitions of the data base into training set and test set.

| pixel representation | | | | feature representation | | | |
|---|---|---|---|---|---|---|---|
| | w.c. | rej. | m.c. | | w.c. | rej. | m.c. |
| test set,θ = 0 | 97.6 % | 0.7 % | 1.7 % | test set,θ = 0 | 97.7 % | 0.4 % | 1.8 % |
| test set,θ = 0.3 | 95.1 % | 3.9 % | 1 % | test set,θ = 0.3 | 96.3 % | 2.6 % | 1 % |

Table 1: Results on the European data base using pixel and feature representation; w.c. = well classified, rej. = rejected, m.c. = missclassified.

| pixel representation | | | | feature representation | | | |
|---|---|---|---|---|---|---|---|
| | w.c. | rej. | m.c. | | w.c. | rej. | m.c. |
| test set,θ = 0 | 93.5 % | 2.4 % | 4.1 % | test set,θ = 0 | 96.5 % | 1 % | 2.5 % |
| test set,θ = 1.2 | 70.9 % | 28.1 % | 1 % | test set,θ = 0.4 | 90.3 % | 8.7 % | 1 % |

Table 2: Results on the U.S. Postal Office data base using pixel and feature representation.

The results on the European data base are almost equally satisfactory for both data representations. When the error rate is further reduced to 0.1 %, we reach a 19 % rejection rate. The recognition rates on the U.S. Postal Office data base vary strongly with respect to the data representation and are only satisfactory when the feature representation is used. This demonstrates impressively the importance of an adequate data representation, especially when the data base exhibits a lot of variations in writing styles and the size of the training set is rather limited.
Our results on the U.S. Postal Office data base using the feature representation appear to be at the state of the art, which is roughly 10 % rejection for 1 % error, for the recognition of handwritten digits without constraints on writing style. However, in comparison to other work [8], our classifier is rather simple and the size of the data base is still moderate.

From the point of view of "real world" application, mere recognition rates are not the only criterion for the choice of a digit recognizer. Whereas the performance of a classifier is measured in terms of its generalization ability, its cost can be measured in terms of complexity of the network (number and type of units, number of trainable connections), of training time and of classification time. The network discussed in this paper may not exhibit the best recognition rates, but it has a very satisfactory "performance-to-cost" ratio.

## 4 - HARDWARE IMPLEMENTATION

An integrated circuit implementing the single-layer network is in the foundry stage [9] at the Laboratoire de Conception de Systèmes Intégrés (INPG, Grenoble). It uses standard 1.2 µm CMOS technology, 24 neurons being implemented on a single chip. Training is performed on a host computer, and the 11,565 weights and the rejection threshold θ of the network are subsequently loaded onto the chips. Table 3 shows

simulation results obtained on the European data base using the pixel representation when weights are stored on 32 bits (floating-point arithmetics), 6 bits and 4 bits (integers) respectively. The threshold $\theta$ was chosen to set the misclassification rate to 1 %. Clearly, there is no substantial decrease in performance when the accuracy of the weights is brought down to 6 bits. Moderate memory requirements and use of binary neurons facilitate greatly the implementation of the network. Classification time for a 256-input pattern is estimated conservatively to 130 µs. The final part of the classification (10 AND operations) is performed by the host computer.

| | w.c. | rej. | m.c. |
|---|---|---|---|
| test set, 32 bits | 96.0 % | 3.0 % | 1 % |
| test set, 6 bits | 95.7 % | 3.3 % | 1 % |
| test set, 4 bits | 93.8 % | 5.2 % | 1 % |

Table 3: Simulation results from the European data base with limited accuracy of the weights.

In a first version of our digit recognizer, all the preprocessing steps (segmentation, line thinning, feature extraction, normalization) are performed by the host computer. In future versions, Digital Signal Processors or dedicated template matching chips, such as the chip described by Graf et al. [10], will perform the multiplications and accumulations of the mask operations.

## 5 - CONCLUSION

We have presented a stepwise procedure for simultaneously building and training a neural network classifier, and its application to a real-world handwritten digit recognition problem. The procedure is based on the idea of simplifying the task assigned to the network by considering a muticlass problem as a set of independent two-class problems, and by using decision trees when necessary. These ideas are implemented in a three-step procedure, resulting in a neural network whose complexity is tuned to the complexity of the classification problem, and which uses binary-state neurons for ease of implementation. The results on the recognition of handwritten digits are comparable to those obtained with networks of much higher complexity. This procedure is not claimed to achieve the best possible results in all cases, but to have a satisfactory performance-to-cost ratio.

**Acknowledgements**
This work was supported in part by BRAIN contract ST2J 0422 and by PRC "Architectures de Machines Nouvelles".

**References:**

[1] S. Knerr, L. Personnaz, G. Dreyfus: "Single-layer Learning Revisited: A Stepwise Procedure for Building and Training a Neural Network", NATO Workshop on Neurocomputing, Les Arcs, France (February 1989); in *Neurocomputing,* F. Fogelman, J. Hérault, eds (Springer, 1990).

[2] M. Mézard, J.P. Nadal: "Learning in Feedforward Layered Networks: the Tiling Algorithm", J. Phys. A **22**, 2191-2203, 1989.

[3] P.E. Utgoff: "Perceptron Trees: A Case Study in Hybrid Concept Representations", Connection Science **1**, 377-391, 1989.

[4] G.Z. Sun, Y.C.Lee, H.H. Chen: "A Novel Net that Learns Sequential Decision Process", in *Neural Information Processing Systems,* D.Z. Anderson, ed. (American Institute of Physics, 1988).

[5] Y. Le Cun, J.S. Denker, S.A. Solla: "Optimal Brain Damage", in *Neural Information Processing Systems,* D.S. Touretzky, ed. (Morgan Kaufmann, 1990).

[6] L. Breiman, J. H. Friedman, R.A. Olshen, C.J. Stone: "Classification and Regression Trees", Wadsworth International, 1984.

[7] S. Knerr, L. Personnaz, G. Dreyfus: "A New Approach to the Design of Neural Network Classifiers and its Application to the Automatic Recognition of Handwritten Digits", in International Joint Conference on Neural Networks IJCNN'91, **1**, 91, 1991.

[8] Y. LeCun, B. Boser, J.S. Denker, D. Henderson, R.E. Howard, W. Hubbard, L.D. Jackel: "Backpropagation Applied to Handwritten Zip Code Recognition", Neural Computation **1**, 541-551, 1989.

[9] P.Y. Alla, G. Saucier, S. Knerr, L. Personnaz, G. Dreyfus: "Design and Implementation of a Dedicated Neural Network for Handwritten Digit Recognition", in *Silicon Architectures for Neural Networks,* M.G. Sami, ed. (Elsevier, 1991).

[10] H.P. Graf, R. Janow, D. Henderson, R. Lee: "Reconfigurable Neural Net Chip with 32-k Connections", in *Neural Information Processing Systems,* R.P. Lippmann, J. E. Moody, D.S. Touretzky, ed. (Morgan Kaufmann, 1991).