Silicon Architectures for Neural Nets, M.G. Sami, ed. (Elsevier, 1991).

Design and Implementation of a Dedicated Neural Network for Handwritten Digit Recognition

Pierre-Yves Alla, Gabrielle Saucier
CSI, Institut National Polytechnique de Grenoble,
46, avenue Félix Viallet
38031 Grenoble, France

S. Knerr, L. Personnaz, G. Dreyfus
Ecole Supérieure de Physique et de Chimie Industrielles
de la Ville de Paris (ESPCI)
Laboratoire d'Electronique
10, rue Vauquelin
75005 Paris, France

Abstract

The automatic recognition of handwritten digits seems to be one of the most promising fields for applications of artificial neural networks; various studies have shown that good recognition rates can be obtained on large "real-world" data bases. This paper presents (i) the design of a network architecture, resulting from a stepwise procedure developed at ESPCI for simultaneously building and training a neural network, intended for the automatic recognition of isolated handwritten digits and (ii) a silicon implementation of that network, using a general-purpose neural circuit architecture developed at CSI.

Introduction

Since the early days of the development of neural networks, the applicability of these systems to automatic character recognition has been extensively studied by various research groups. Clearly, this field is among the most promising for applications of neural networks, and various prototypes are operating satisfactorily.

The present work reports a collaboration between two research groups with different but complementary scopes:

- the Laboratoire d'Electronique of ESPCI, which has been investigating extensively the automatic recognition of handwritten digits by neural networks, and has developed a new procedure which simultaneously builds and trains a neural network classifier; in the present paper, we describe the procedure and its application to the recognition of handwritten digits, together with the network structure which was the outcome of the procedure,
- the Laboratoire de Conception de Systèmes Intégrés (CSI), where a general-purpose silicon architecture has been developed which allows the implementation of virtually any neural network structure; this architecture is used to implement the network proposed by ESPCI.

The circuit is intended to perform the automatic recognition of handwritten isolated digits, *normalized* to 256 pixels with 16 gray levels. Hence, the preprocessing steps (which may include segmentation, in the case of unconstrained postal codes, for instance) are reduced to well-known operations which can be carried out by various classical techniques that shall not be described here.

The stepwise building and training procedure

At present, training algorithms, such as backpropagation, start from a fixed network architecture which has to be determined in advance by guesses, possibly guided by a priori knowledge on the problem, leading to constrained structures [Ham90, LeC89]. In addition, backpropagation tends to be time-consuming even when used to train networks of small size. The basic idea of the stepwise procedure (and of procedures in the same spirit suggested by other authors [Méz89, Utg89]) is to produce a network structure which is matched to the complexity of the classification problem it is intended to solve, by simultaneously building and training the neural network. The resulting network has a *single layer of adaptive connections* between the inputs and the first layer of *binary neurons*, and additional layers which perform boolean operations. Therefore, training is fast since it is performed on one layer of connections only, and the hardware implementation is relatively straightforward since the neurons have binary outputs.

The building and training procedure consists of three steps:

- in a first step, the procedure attempts to separate the examples of each class linearly from all others; this results in successful neurons, which are kept, and in unsuccessful neurons, which are discarded;
- in the second step, the procedure attempts to separate linearly the *pairs of classes* which were not separated during the previous step; again, the resulting successful neurons 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) are separated pairwise by *piecewise linear* decision surfaces.

Piecewise linear separation surfaces are generated as follows [Kne89]: starting out

with one neuron which divides the input space into two regions, there exist misclassified examples in at least one of these regions (otherwise, the two classes would have been separated successfully in step two of the procedure). Then, in each region containing misclassified examples, a neuron is generated to separate the well classified examples from the misclassified examples. Iterating this procedure, the problem is split up as in a classification tree [Bre84], resulting in a decision surface of increasing complexity. Neurons are added until cross-validation shows that the generalization ability is no longer improved. In order to simplify hardware implementations, the tree structure can be transformed into a simple network by adding two layers of boolean functions.

Steps (1) and (2) make use of the generalized delta rule [Kne89], while step (3) uses the Perceptron Pocket algorithm [Gal86]. In the first two steps, training requires neurons with sigmoidal nonlinearities, whereas *binary* neurons are used in the classification phase. The last step might be carried out with multilayer networks trained by backpropagation to separate the remaining classes (in that case, the separation surfaces would not be piecewise linear).

A Handwritten Digit Recognizer

The above procedure was applied to build and train a network for handwritten digit recognition. The data base used for training and testing the network consists of 8700 unconstrained handwritten digits (870 per class). The binary images from an optical scanner were normalized to 16 x 16 pixels with 16 gray levels. Some examples of the data base are shown on figure 1.

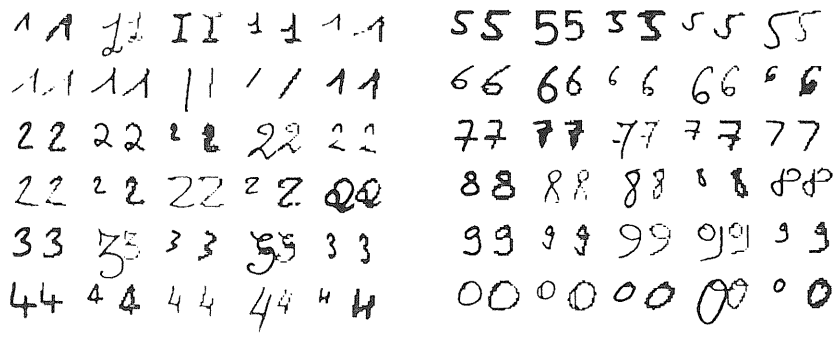

Figure 1: Samples from the data base used for generating the network. For each pair of digits, the left one is the original and the right one is the normalized digit.

It turned out that the stepwise training procedure stopped after the second step, thereby indicating that the classes of the training set are pairwise linearly separable.

Thus, the resulting architecture of the network is very simple (figure 2): a single layer of 45 neurons is fully connected to the 256 four-bit inputs, representing the 16x16 pixel images. The final decision is made by 10 AND gates, connected to 9 neurons each.

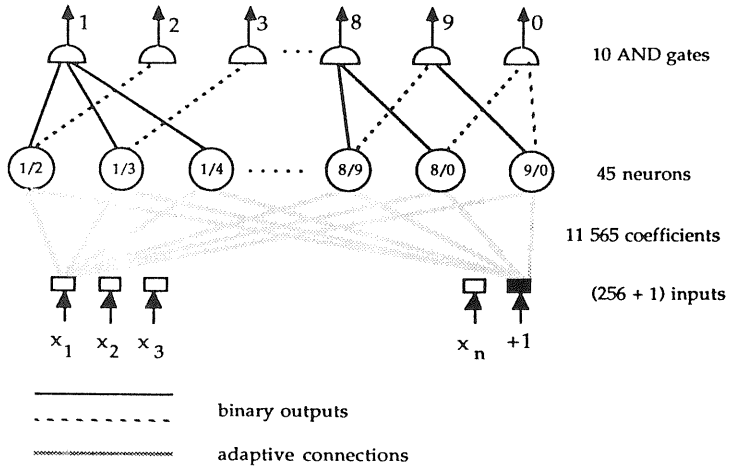

Figure 2: Network architecture

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 (non classified) 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 absolute value of a potential indicates an ambiguous situation. Figure 3 illustrates the 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 Θ; the two binary outputs (i/j)j and (i/j)i are then:

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

The final decision of the network is as follows:

- if (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.

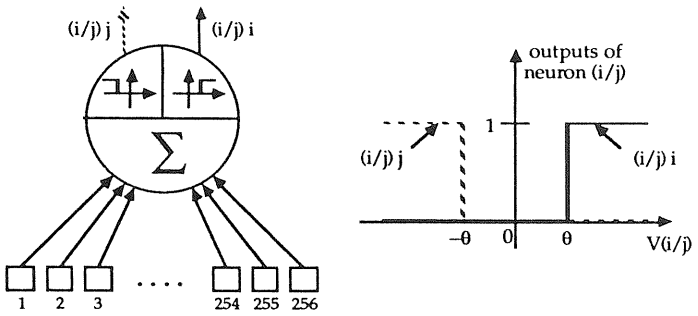

Figure 3: Neuron (i/j) and rejection mechanism

Table 1 shows simulation results obtained with the network which was trained on half of the data base and tested on the other half of the 8700 digits. The first row gives the results on the training set, indicating that the data was learned almost perfectly. The results of the second, third and fourth row were obtained on the test set with coefficients stored on 32 bit (floating-point arithmetics), 6 bits and 4 bits (integers) respectively. The threshold Θ was chosen to set the misclassification rate to 1 %. Clearly, there is no substantial decrease in performance when the precision of the coefficients is brought down to 6 bits. The moderate memory requirements facilitate greatly the implementation of the network.

	w.c.	rej.	m.c.
training set	99.5 %	0.3 %	0.2 %
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 1: Simulation results on a 8700 digits data base
(w.c. = well classified, rej. = rejected, m.c. = misclassified)

We also compared these results to those obtained by multilayer perceptrons, trained with backpropagation. Using an unconstrained architecture with one hidden layer and an optimized number of hidden units, the best results came close to those shown in Table 1.

A general purpose implementation

A neural network architecture in the spirit of a dedicated silicon compiler has been developed at the LCSI [Oua89]. This architecture is based on the replication of a basic computing element (the "neural processor"). Training is performed on a host computer and the coefficients of the network are loaded onto the chip. In the following, the implementation of the previously described network with the LCSI architecture is presented. First, the structure of an individual neuron is described and then the network architecture is discussed.

The neuron

The main idea is to design a single neural processor, with an autonomous control, a local memory to store the coefficients, and identification registers which contain the necessary information about the global architecture of the network.

The processor itself is simple and can be divided into two parts. The first part computes the potential, and the second part performs the nonlinear function and the rejection mechanism, based on the potential $V_{(i/j)}$ and on the rejection threshold Θ which can be loaded from outside. The mutiplication of the inputs by the corresponding coefficients is performed using the classical Booth algorithm. The adder/subtractor required by this algorithm is also used to accumulate $V_{(i/j)}$ and to perform the rejection thresholding.

Since the two outputs of the neuron are binary, they are not transferred through the busses, but they are directly transmitted to the host computer from the controller. Therefore, there is no need for an output register. Figure 4 shows the design of a neural processor.

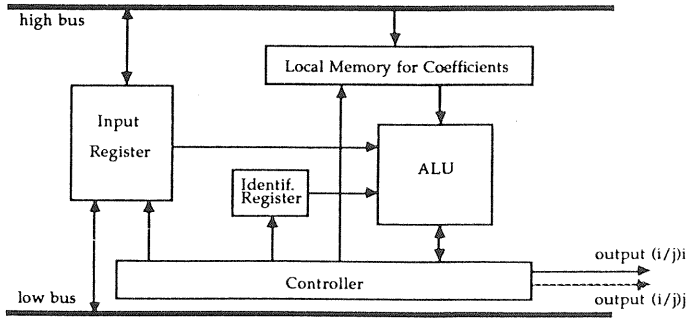

Figure 4: The neural processor for classes (i/j)

Data is encoded on 4 bits (positive integers), and coefficients are encoded on 6 bits. Simulations have shown that the potential never exceeds the value of 5000; therefore, the accumulator is only 13 bit wide. The busses must be 6 bit wide, to be able to load the coefficients into the memories.

The network

The CSI architecture [Oua89] has been designed to allow the implementation of a wide variety of neural network structures. In order to reduce connection costs, the first layer of the network is organized in 4 rows of 12 neurons each (figure 5), each row lying between two bus segments. As mentioned above, only 45 neurons are required for the first layer, but the 48 neuron structure allows a more regular design. The first layer is folded into two columns of bus segments in order to ease the circulation of the data. The principle of operation is the following: 8 inputs are entered in parallel, and all the neurons attached to the same bus segment operate on the same input, at a given instant of time. At the next time step the data circulates via the neurons in the direction indicated by the arrow. Within 8 time steps each neuron operated on each of the 8 inputs; then the next 8 inputs are entered. Thus, the 256 pixels of the pattern are input in 32 packets.

This architecture may not be optimal for the discussed application, but the constraints that it imposes on the design are easily manageable. The original structure was supposed to implement a multilayer network with sigmoidal nonlinearities. Since, in the present case, the outputs of the neurons are binary, the busses are not needed to broadcast the outputs to the layer which performs boolean AND functions; therefore the busses will only be used to supply the neurons with the data and to load the memories. As a result, the busses need not be managed once all the data has been loaded.

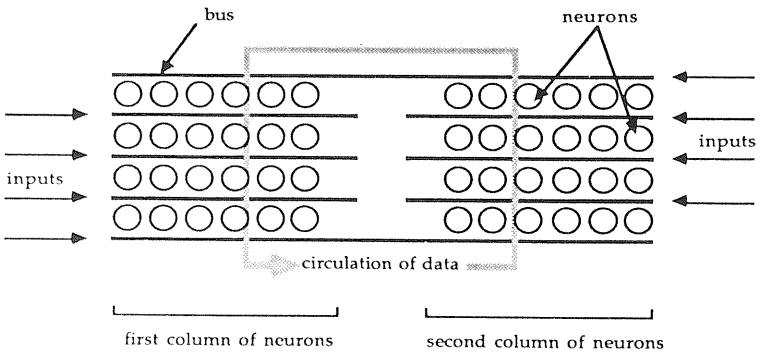

Figure 5: The stucture of the first layer and data circulation therein

The second layer, which is composed of 10 AND gates with 9 inputs each, is not implemented using the "neural processor". In a first version of the classifier, the 1-bit outputs are broadcasted to the host computer, where the final part of the classification will be implemented in software.

The chip

The circuit has been designed using VLSI Technologies Inc. software. The "neural processor" can be divided into three parts as follows:

- the operative part, which contains the ALU, the inputs and the identification registers. This part is generated with a Data Path Compiler.
- the controller, described as a state machine, and generated by the State Machine Compiler.
- the local memories of 256 6-bit words.

The size of one neuron is smaller than 3 mm^2, with a 1.2 micron technology. This allows an implementation of 24 neurons on one chip. Thus, the complete first layer of the network fits on two chips. ROMs instead of RAMs could be used for the local memories in order to implement the whole network on one chip. In this case the coefficients of the neural network are fixed once and can not be changed when dealing with different data bases.

The time needed to classify one input pattern can be estimated to be 2600 cycles, with a theoretical minimum cycle time of about 50 ns. Therefore, the on-chip recognition time for one digit can be estimated to be approximately 130 µs.

Conclusion

A dedicated neural network for handwritten digit recognition has been designed and is being implemented in silicon. The final network will be a set of two chips with 45 binary neurons, performing the classification in parallel. The method requires a minimal amount of preprocessing.

This work showed:

(i) the effectiveness of the stepwise building and training procedure in producing simple but efficient networks adapted to real-world classification problems,

(ii) the flexibility of a general-purpose architecture, in the spirit of a "neural" silicon compiler, for the implementation of a dedicated network architecture.

A hardware implementation of the preprocessing tasks will be considered in the future.

Acknowledgments: this work was supported in part by ECE contract ST2J 0312C, by the G.CIS of CNRS, and by the PRC "Architectures de Machines Nouvelles".
The authors are grateful to D. Urbani for simulation work on the character data base.

References

[Bre84] L. Breiman, J. H. Friedman, R.A. Olshen, C.J. Stone: "Classification and Regression Trees", Wadsworth International, 1984.

[Kne89] 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).

[Gal86] S.I. Gallant: "Optimal Linear Discriminants", 8th International Conference on Pattern Recognition Vol. 2, 849, Paris, France, 1986.

[LeC89] 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.

[Ham90] J.B Hampshire, A.H. Waibel: "A Novel Objective Function for Improved Phoneme Recognition Using Time-Delay Neural Networks", IEEE Trans. on Neural Networks **1**, 216-228, 1990.

[Oua89] J. Ouali, G. Saucier: "A Distributed Architecture for Neural Networks Based on a Neural Processor", 2nd International Workshop on Neural Networks and their Applications, Nîmes, France, November 89.

[Méz89] M. Mézard, J.P. Nadal: "Learning in Feedforward Layered Networks: the Tiling Algorithm", J. Phys. A **22**, 2191-2203, 1989.

[Utg89] P.E. Utgoff: "Perceptron Trees: A Case Study in Hybrid Concept Representations", Connection Science **1**, 377-391, 1989.