A VERY HIGH DENSITY VLSI IMPLEMENTATION OF THRESHOLD NETWORK ENSEMBLES (TNE)

A. Bermak

Hong Kong UST,
EEE Department,
Kowloon, HK, SAR, China

D. Martinez[*]

LORIA-CNRS
BP239 Vandoeuvre,
Nancy, 54506, France.

ABSTRACT

This paper describes a hardware implementation of threshold network ensembles (TNE) for classification applications. We first describe the algorithm and compare its performance with those of individual classifiers such as binary neural network and support vector machine (SVM). The effect of limited precision on the performance of threshold network ensembles is also investigated. The proposed multi-precision architecture is then mapped into a scalable systolic architecture implemented first on a single VLSI chip. The modularity and the easy programability of the basic chip has made possible the extension of the architecture to a low cost multi-chip solution. We propose a 3D packaged circuit in which 12 basic chips have been integrated into a very compact volume of $(2 \times 2 \times 0.7)cm^3$. Successful operation of the 3D prototype is demonstrated through experimental test results of the chip.

1. INTRODUCTION

Classification is one of the most important tasks in pattern recognition intelligent systems. Different classification schemes have been proposed in the literature [1]. The appropriate solution, for a given problem, is based upon the experimental assessments of the different possible schemes. Eventhough in such experimental assessments, one of the classification schemes would yield the best performance, the sets of patterns miss-classified would not necessarily overlap. This suggests that different classifiers offer complementary information about the patterns to be classified. Consequently, combining a set of classifiers to build an ensemble is an advanced pattern recognition technique which has gained increasing attention within the machine learning community [2]. Creating network ensembles is realized using resampling techniques, such as Bagging [3] and Boosting, to obtain different training sets for each of the individual classifiers. The resulting combined classifier is generally more robust and accurate as compared to any individual classifier making up the ensemble. Unfortunately ensembles do suffer from a number of shortcomings mainly due to their large requirements in terms of storage memory and computational power [2]. Although all network ensembles implementations reported in the literature are realized using software simulations, only hardware realization can fully meet the very challenging requirements of real-time processing. However, when implementing network ensembles with a dedicated hardware a number of issues need to be addressed. These include (i) the reconfigurability of the hardware and its programability in order to support different classification problems and (ii) performance degradation due to limited precision. In this paper, we propose a very high density chip that we believe can address these issues and can meet the computational requirement of Network Ensembles. Individual classifiers are based on binary neural network and decision trees and are referred to as *Threshold networks*. The proposed architecture is mapped into a scalable systolic architecture implemented first on a single VLSI chip. The modularity and the easy programability of the basic chip has made possible the extension of the architecture to a multi-chip solution. Section 2 of the paper describes the architecture of the TNE and its simulation and comparison with single classifiers for different precision requirements. Section 3, describes the hardware architecture of the multiprecision 3D chip together with the experimental results and the chip performance. A conclusion is discussed in section 4.

2. ARCHITECTURE OF TNE

Figure 1 shows the block diagram of a network ensemble for which the output is given by:

$$Y = \text{sgn}(\sum_{i=1}^{n} \alpha_i f_i(x)) \quad (1)$$

where n is the number of individual classifiers and $f_i(x)$ is the ± 1 output of the i^{th} individual classifier within the ensemble. sgn is the *signum* function (sgn$(x) = 1$ if $x \geq 0$ and -1 otherwise) and $\alpha_1 \cdots \alpha_n$ are the weights associated with the individual classifiers. For an ensemble created by

[*]This work was initiated while the authors were at LAAS-CNRS.

bagging [3], these weightings are equal, $\alpha_i = \frac{1}{n}$ for $i = 1 \cdots n$, so that the output is given by a simple majority vote. Thus, Y= +1 or -1 if a majority of individual classifiers gives an output +1 or -1, respectively.

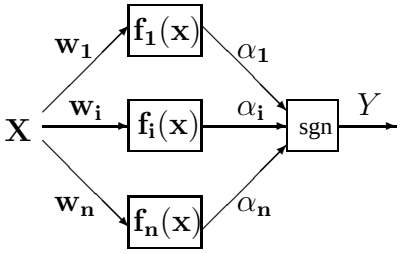


Fig. 1. Network Ensemble. The $f_i(x)$ are individual classifiers. X is the input vector, W_i is the weight matrix and the α_i are the weightings for each classifier.

In such ensembles, simple individual classifiers that use threshold logic units (TLUs) such as binary neural networks or decision trees are often used. They offer the advantage of being easy to train and implement. Indeed, there exist numerous constructive algorithms for simultaneously building and training binary neural networks [4] or decision trees [5]. A binary neural network can also be represented as a decision tree and vice versa. This can be evidenced through the example shown in Figure 2.

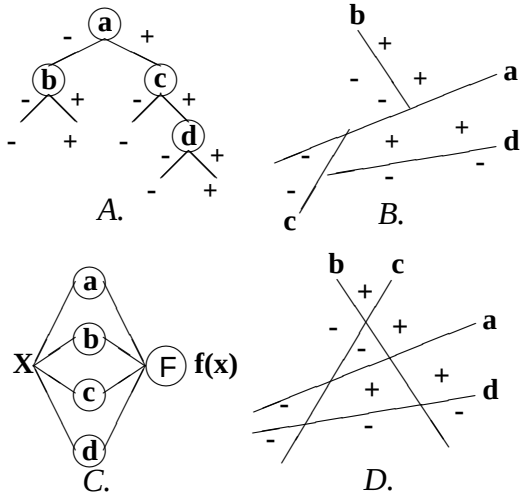


Fig. 2. Equivalence between decision trees and binary neural networks (threshold networks). (A.) and (C.) are two examples of a tree and a threshold network respectively and (B.) and (C.) are their respective partition of the input space. Each node of a decision tree corresponds to a separator hyperplane and the leaves of the tree correspond to a given class. Each class can be represented by a logical function that combines a set of leaves. In our example $class_+ = \bar{a}b + acd$.

The decision tree shown in Figure 2.A implements a classifier that discriminates between two classes (denoted + and - in Figure 2.B). Each node in the tree is a TLU implementing a linear discriminant and each leaf is associated to a given class. Classifying an input pattern then reduces to a sequence of binary decisions, starting from the root node and ending when a leaf is reached. Each class can then be represented by a logical function F that combines the binary decisions encountered at the nodes and a decision tree can thus be considered as a binary neural network having a hidden layer of TLUs followed by a logical function as shown in Figure 2.C and 2.D. Therefore, each individual classifier i in Figure 1 is considered as a threshold network, i.e. a network of TLUs followed by a logical function F_i

$$f_i(X) = F_i(\text{sgn}(W_i\ X)) \tag{2}$$

where W_i is the weight matrix for classifier i, X is the input vector and the function sgn(x) is the *signum* function that applies to every component of vector X. To evaluate the performance of threshold network ensembles as described by eqs. (1) and (2), we performed 2-class discrimination experiments on three datasets : 'hepatitis' and 'ionosphere' from the UCI repository [6] and 'odor' from the NOSE project [7]. The two formers have been previously investigated by other researchers [8]. The latter has been developed by us for training a classifier capable of discriminating between two odors (ethanol and butanol) and consists of gas sensors responses recorded at various concentrations of the odors. Research work suggested that ensembles with ten members (classifiers $f_i(x)$) are adequate to improve the classification performance [9]. Thus, ensembles of ten threshold networks were created. We used bagging [3] so that the weightings in eq. (1) are equal, $\alpha_i = \frac{1}{n}$ for $i = 1 \cdots n$. Each individual threshold network within the ensemble was created using the following procedure : First, a decision tree is trained using OC1 without pruning [10] and second the trained decision tree is transformed into a network of TLUs as described by eq. (2) and represented in Figure 2C. The logical function F_i was automatically extracted from the tree structure. Table 1 reports the performance of such ensembles in comparison to the one of single threshold networks and support vector machines (SVM) [11]. For these datasets, threshold network ensembles were always more accurate than any of their individual component classifiers, which agrees with previous findings [9]. Moreover, they outperformed SVMs in two datasets over three.

Dataset	Single Threshold Network	Ensemble of 10 Threshold Networks	SVM
hepatitis	78.75	*88.75*	83.75
ionosphere	89.46	*93.16*	89.74
odor	89.39	93.18	*93.94*

Table 1. Leaving-one-out accuracy (in %) for the different datasets. For each dataset, performance of the best classifier is indicated in italic. See text for explanation on the procedure used for creating individual and ensemble of threshold networks. For SVM, polynomial kernels have been used and the degrees of the polynomial have been adjusted separately to get the best performance on each individual dataset.

When implementing threshold network ensembles with hardware of limited precision, the sensitivity to weight perturbation may result in performance degradation. It is therefore very important to study carefully the precision requirements for the classification problem at hand. Synaptic weights and inputs are then coded with the minimum required precision without affecting too much the classification performance. We have evaluated the effect of weight precision on the performance of threshold network ensembles for the three datasets used above. After training, the weight vectors of each individual threshold network were normalized and uniformly quantized with b bits of precision by assigning $N = 2^b$ uniform intervals over the range (-1, +1). Table 2 reports the performance of threshold network ensembles with weights quantized with 16, 8 and 4 bits. In order to maintain acceptable performance, 16 bits of precision are sufficient for all the datasets. However, the required precision depends on the problem at hand (16 bits for 'hepatitis' and 'odor' against 8 bits only for 'ionosphere'). The use of a wordlength larger than the required precision (for example 16 bits for 'ionosphere') results in an inefficient usage of the hardware resources (slower processing and higher power consumption). Since the required precision depends on the problem at hand, we have chosen to implement threshold network ensembles in hardware with multiprecision. This permits to exploit efficiently the hardware resources available and hence facilitate the implementation of reasonable size network ensembles.

Dataset	16 bits	8 bits	4 bits
hepatitis	*87.50*	73.75	66.25
ionosphere	92.59	*92.31*	53.84
odor	*93.18*	90.91	68.94

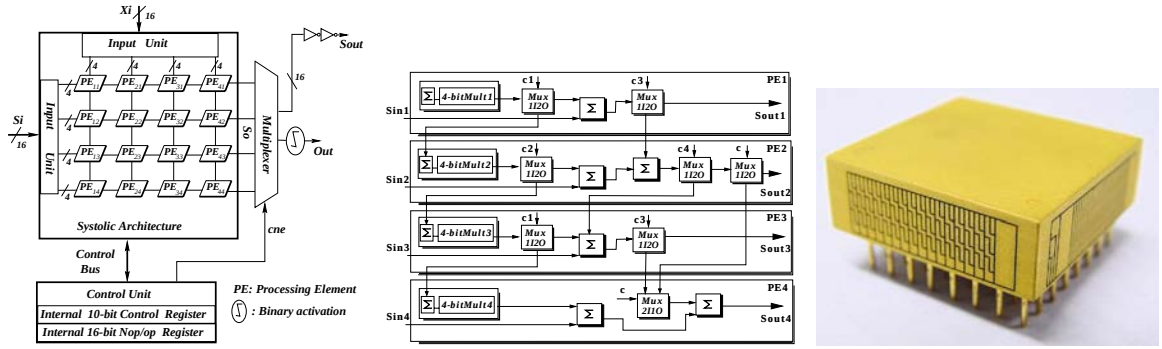
Table 2. Leaving-one-out accuracy (in %) of ensembles of ten threshold networks with respect to the weight precision (in number of bits). Performance for the required precision is indicated in italic.

3. HARDWARE IMPLEMENTATION

Threshold network ensembles require only TLUs and are very suitable for a VLSI implementation. The threshold function is indeed easy to implement in digital and this results in significant silicon area saving as compared to sigmoidal or radial basis functions used in multilayer perceptrons or RBF networks and implemented through area consuming look-up tables. This simplification results in very compact arithmetic units, and makes the prospect of building up VLSI chips implementing threshold network ensembles particularly promising for real time decisions.

The basic building block chip is based on a scalable 2D systolic array architecture. This array consists of 4×4 Processing Elements (PE) as shown in Figure 3.A. The array could be configured to perform either a weighted sum (Sout) or a TLU units (Out) . The architecture can be expanded horizontally in order to realize a network with more inputs as well as vertically in order to increase the number of TLUs. The buses Si and Sout are systolic Input/Output buses used to interface between basic VLSI chips when higher number of PEs is needed. Figure 3.B shows the internal schematic of the configurable multi-precision arithmetic unit. The 16-bit arithmetic unit was built-up as a four 4-bit processors wired together using a set of multiplexers allowing to change the hardware connections between two adjacent rows of cells in order to obtain a weight precision of 4, 8 or 16-bit. Depending on the selected precision different topologies of networks can be configured. $P_b p \times q$ stands for a network topology with b bits of precision, p inputs and q TLUs. For a 4-bit precision three configurations are possible namely: $P_4 16 \times 4$, $P_4 8 \times 8$ and $P_4 4 \times 16$. For an 8-bit precision two configurations are possible: $P_8 4 \times 8$, $P_8 8 \times 4$ and only one configuration is possible for a 16-bit precision: $P_{16} 4 \times 4$. Larger networks (more inputs per TLU) are obtained by bypassing the activation function and connecting the partial weighted sum across different chips.

The basic chip has been fabricated using standard cell $0.7\mu m$ CMOS technology. After designing the chip and successfully testing it, four dies were mounted on a single Multi-Chip-Module (MCM). Each MCM implements a fully configurable systolic array constituted of 16×16 16-bit PEs. The 3D packaging technology referred to as MCM-V [12] was used to realize the 3D chip. After test of the MCMs, the selected ones are stacked, one above the other and encapsulated in epoxy resin. After a global metalization of the cube, a laser is then used to pattern the surface so that vertical wire tracks are formed on the cube [12]. Interconnections between the layers are realized on the sides of the module. Each step in the fabrication of the 3D module uses a standard and well-characterized technological process. As a consequence, the MCM-V technology is relatively low cost [12]. Figure 3.C shows a photograph of the final 3D chip. The module includes four substrates layers with 4 chips on each of the three top levels (12 chips in total). The Bottom substrate is fully dedicated to the report of the vertical connections to an external PGA package. The size of the final module is $2 \times 2 \times 0.7 cm^3$. This represents at least 25% of the volume of an advanced PCB implementation. The 3D chip has been successfully tested for all configurations of precision and TLU topology. It presents a loading time of $10.8\mu s$. This value corresponds to the time required to load the synaptic words (768 words of 8 bits) and the control sequences. The processing time for a single recall operation varies from $611ns$ to $2111ns$ depending on the selected topology of the threshold network and precision. Figure 4 shows the experimental output from the chip which corresponds to a full recall cycle for the topology $P_4 4 \times 192$. A 4×4 weight matrix W was loaded into each chip within the 3D prototype. A test vector X^T was also fed serially to the chip from the least significant bit to the most significant



(A.) (B.) (C.)

Fig. 3. (A.) Internal architecture of the VLSI chip. (B.) Internal schematic of the multiprecision arithmetic unit. (C.) Photograph of the 3D chip.

one. The values of both W and X^T were chosen so that the output of two adjacent rows of the systolic array would have opposite sign and hence the TLUs outputs (Out signal) would oscillate at $f/2$. A chip within each MCM has been selected for the purpose of this test. The first neuron's output is obtained after 8 clock cycles of the load acknowledgment signal (lw_{out}). The 192 TLU outputs are obtained in only 23 clock cycles using only 12 physical PINs. Each output would generate sequentially the results corresponding to 16 TLUs as shown in Figure 4. This corresponds to a very high level of parallelism realized with very limited physical outputs.

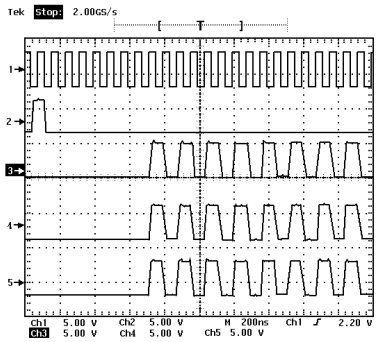


Fig. 4. Experimentally measured sequence for the loading acknowledgment signal Lw_{out} and the output of the different neurons for topology: $P_4 4 \times 192$. Ch1-Ch5 representing the clock signal, the acknowledgment loading, the TLUs outputs from the first second and third chip respectively.

4. CONCLUSION

In this paper we have presented the algorithm, 3D circuit implementation and experimental test results of neural network classifiers using threshold network ensembles. The algorithm is based on combining a set of elementary binary classifiers in order to achieve higher classification performance. We have demonstrated, for different datasets, that such classifiers were always more accurate than any of their individual component classifiers. Moreover threshold ensembles outperformed SVMs in most tested problems. We have also investigated the performance degradation due to limited precision of the synaptic weights and proposed to implement threshold network ensembles in a multi-precision hardware (4/8/16-bit). The 3D chip represents a very powerful neural processor including 16-bit 192 PEs implementing 768 digital synapses. It exhibits a significant speed of 1.25 GCPS as compared to other digital NN hardware.

Acknowledgments

Thanks are due to T. Doconto for wire-bonding the chips and to C. Val, 3D-plus for manufacturing the 3D module.

5. REFERENCES

[1] R. O. Duda; P.E Hart and D. G. Stork, "Pattern Classification," Wiley-interscience, 2001.

[2] T. G. Dietterich, "Machine Learning Research: Four Current Directions," AI Magazine, 18(4), 97-136, 1997.

[3] L. Breiman, "Bagging predictors," Machine Learning, 24(2), pp.123-140, 1996.

[4] J. M. T. Moreno and M. B. Gordon, "Efficient adaptive learning for classification tasks with binary units," Neural Computation, 10(4), pp.1007-1030, 1998.

[5] L. Breiman, J. H. Friedman, R. A. Olshen and C. J. Stone, "Classification and regression trees," Wadsworth Int. Group, Belmont CA, 1984.

[6] C.L. Blake and C.J. Merz, "UCI Repository of machine learning databases, " Irvine, CA: UC, DICS, 1998.

[7] "Neuromimetic Olfactory SEnsing (NOSE), " "http://www.loria.fr/ rochel/nose "

[8] E. Bauer and R. Kohavi, "An Empirical Comparison of Voting Classification Algorithms: Bagging, Boosting, and Variants," Machne Learning, 36, pp.105-139, 1999.

[9] D. Opitz and R. Maclin, "Popular Ensemble Methods: An Empirical Study," Journal of Artificial Intelligence Research, 11, pp.169-198, 1999.

[10] S.K. Murthy; S. Kasif and S. Salzberg, "A System for Induction of Oblique Decision Trees," Journal of Artificial Intelligence Research 2, pp. 1-33, 1994.

[11] C. Cortes and V. Vapnik, "Support vector networks," *Machine Learning*, 20, pp 1-25 , 1995.

[12] S.P. Larcombe, et al., "Electronic systems in dense three-dimensional packages," *Electronics Letters*, Vol.31, N.10, pp.786 -788, June 1995.