1994-11-09

Invited talk 1994 WNN/FNN, Washington DC, December 1994

An implementation of the Zero Instruction Set Computer (ZISC036) on a PC/ISA-bus card

Å. Eide[1)], Th. Lindblad[2)], C.S. Lindsey, M. Minerskjöld,
G. Sekhniaidze[3)], and G. Székely[4)]

Royal Institute of Technology
Department of Physics - Frescati, Stockholm, Sweden

Abstract: An implementation of the new IBM Zero Instruction Set Computer (ZISC036) on a PC/ISA-bus card is reported. This circuit has 36 processing elements of a type similar to that of Radial Basis Function or RBF-like neurons. It is a highly parallel and cascadeable building block with on-chip learning capability, and is well suited for pattern recognition, signal processing, etc. A card with two ZISC036 was built and tested with a noisy character recognition "benchmark". Some future implementations and ideas are presented.

1. Introduction

We describe the first implementation of the IBM Zero Instruction Set Computer (ZISC036) on a PC-486 ISA-bus card. This chip has a RBF-like [1-2] neural network topology and a RCE learning algorithm embedded on-chip [3]. Although this learning concept involves simple commitment of training vectors (usually called prototypes) to memory and adjustment of scalar weighting factors, such networks have been shown to define classification boundaries of complicated clustering in multidimensional space [3- 7]. These are features that make it well suited for image and signal recognition and processing, as well as other tasks, although generalization power *may* be smaller than that of feedforward networks trained by backpropagation. Generally, RBF networks also require more neurons than many other types of networks. This may be overcome by the easy way of cascading several ZISC036 chips (each with 36 neurons) as well as future chips in this series. Also, the comparison of input vectors to stored prototypes involves only subtractions and additions. Hence, without time consuming multiplication operations, the chip can be quite fast.

In this paper, we will first briefly review the concept of the Radial Basis Function (RBF) type of neural network. The chip, hardware and the software implementation, will then be discussed in sections 3, 4, and 5, respectively. Finally, the results of some tests will be presented together with a summary and a discussion of future areas of applications.

[1)] Permanent address: Ostfold College, Halden, [2)] Corresponding author, email: LINDBLAD@VANA.PHYSTO.SE
[3)] Permanent address: Georgian Academy of Sciences, Tbilisi, [4)] Permanent address: Hungarian Academy of Sciences, Debrecen

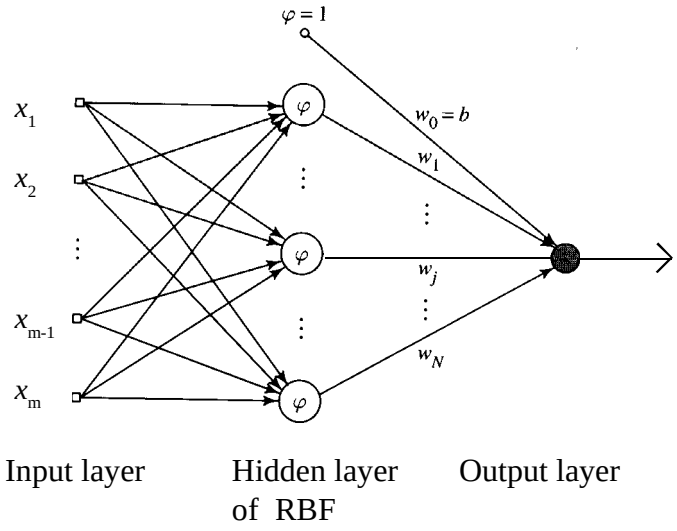

Figure 1. Radial basis function (RBF) network. The network is of the feedforward type with one hidden layer.

2. Radial Basis Functions

Classification problems require the labeling of inputs according to where they lie in the input space. For example, a given input vector to a network with 64 individual input elements represents a single point in a 64 dimensional space. Using a feedforward network involves training with an algorithm like back-propagation to construct image points (by the hidden layer) that are linearly separable, by one or more hyperplanes, and thus making the original clusters separable (ref [1-4]).

Instead of hyperplanes, another method to identify points as members of a given cluster involves constructing a function, or sum of functions, that is non-zero inside a cluster and zero outside of it. Such a function could be expressed generally as

$$\varphi(X) = \varphi(|X - P|, A), \qquad (1)$$

that depend on a scalar quantity proportional to the difference between the input vector and a point in the cluster (e.g. the vector distance) and a parameter A that characterises the extent of the function (e.g. the sigma of a Gaussian). Such a function is "radial" since it depends only on the radial distance from a center point. Here $\varphi(X)$ would be non-zero when X is within a given (e.g. Manhattan) distance of P, and zero otherwise (or below a threshold). The normal procedure is to use simple functions, like Gaussians or signum functions, for $\varphi(X)$ and to characterize a complicated cluster shape with a sum of such functions. The only information usually available, however, are N labeled points in a training set. So the grand function could be generalized as

$$F(X) = \sum [w_i * \varphi(|X - P_i|, A_i)], \qquad (2)$$

(with the summation running from $i = 1$ to N) where the w_i and A_i are parameters to be determined that give the "best" approximation to the "true" $F(X)$. An alternative view of this is as a transformation of the X into a new N-dimensional space where the clusters are linearly separable. Then the $\varphi(x)$ are

the ”basis” of this new space and thus referred to as radial-basis functions (RBF).

Figure 1 shows a feedforward network implementation of such a network. Each middle layer neuron implements a $\varphi(|X - P_i|)$ function, where the P_i prototype vector is stored in the neuron. The output layer neurons, each representing a given class, are linear sums of the $\varphi(X)$ functions.

There are many methods described in the literature, see for example ref. [4-7], to build the $F(X)$ from RBF's. Such methods often try to insure that the resulting function is ”smooth”, using some type of regularization procedure, and so can interpolate well between the training points. Another important goal is to find the minimum number of RBF's needed to separate the clusters.

Many of these methods are basically making fits to a given fixed set of P_i points while constrained by a regularization term. An alternative approach is to ”build” the function as new data is input (e.g. in a control application the data may not be known a priori.) The RCE method (ref. [3]), for example, examines each new vector and builds the sum accordingly. In this method, the W_i's are fixed to 1.0 between the output neuron and the middle layer neurons that correspond to that output neuron class and 0.0 otherwise. The RBF's are simple signum functions (i.e. step functions), with A_i the radius of the function. An input belongs to the $\varphi(X)$ class if $\varphi(X)$ is non-zero. During training, if only one or more neurons of the correct class fires then nothing is done. If no current middle layer neuron fires, a new middle layer neuron is created with the input vector stored as the prototype P for that neuron and having a default radius. If a wrong neuron fires, its radius is made slightly smaller than the distance to the current vector. Such a method can build separating functions with even very complicated cluster shapes, assuming enough middle layer neurons are available.

3. The Zero Instruction Set Computer (ZISC) chip

Referring to the ZISC036 data book [8], the circuit is the first element in a series from IBM of fully integrated implementations of RBF-like neural networks. It has been presented in some detail by LeBouquin [9]. However, for the convenience of the reader we summarize the properties of the circuit below.

The ZISC036 has been designed for cost-effective recognition and classification in real-time. This is achieved by a high degree of parallelism. Thus each neuron has a register file for prototype storage, as well as a unit for evaluation of distances. Although the ZISC036 chip contains only 36 neurons, it is easily cascadable and the total number of neurons in the network is not limited (cf. the Ni1000 chip[1)]). The upper limit of the number of categories is equal to 16 K.

The distances can be calculated according to two selectable norms (cf. above), or

$$L_1 = \sum |V_i - P_i|, \qquad (3)$$

where V_i and P_i represents the input vector and the stored prototype respectively. The summation runs

[1)] The Ni1000 RBF-chip [7] has 256 inputs, 1024 hidden and 64 output neurons. It has an on-chip micro-controller and the 1K prototypes are stored in a 1.3 Mb flash EPROM. At 40 MHz this 3.7 million transistor chip executes 20B 5-bit integer subtract and accumulate operations per second and 160 M fpc/s.

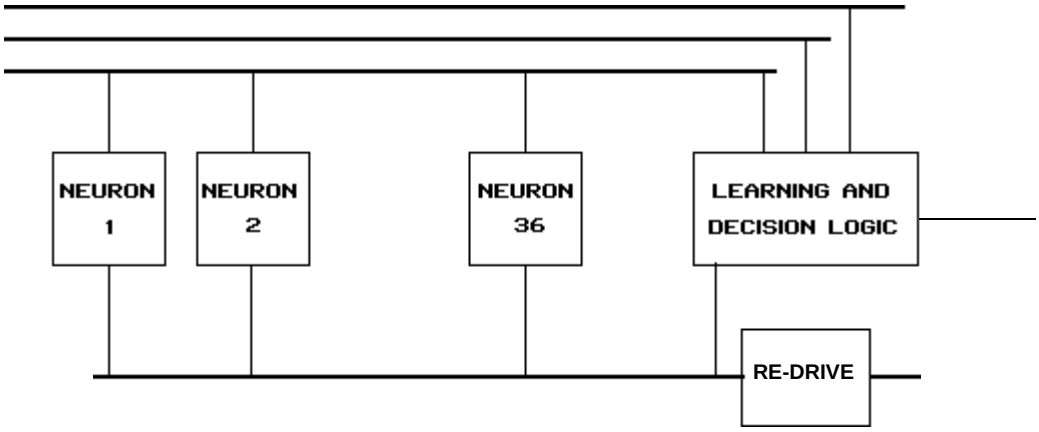

Figure 2. ZISC036 block diagram. The top part shows (from top to down)the address (6-bit), control (9-bit) and I/O data (16-bit) buses; the lower right part includes the communication (16-bit) bus. All buses are directly connected when chips are cascaded. The 4-bit decision bus (far right) allows the use of the ZISC036 in stand alone mode.

from 1 to 64 and each component of the vector is coded as an 8-bit number. Alternatively, we can have

$$L_{sup} = \max |V_i - P_i|, \qquad (4)$$

which is referred to as a "hypercube field". In both cases the distance calculations are carried out using 14-bit precision. The components of the vectors are fed in sequence to each neuron and processed in parallel. This means that if the ZISC036 is operated at 20 MHz, 64 components can be fed and processed in 3.2 µs. The evaluation is obtained 0.5 µs (or one clock cycle) after the feeding of the last component, which corresponds to a quarter of a million evaluations per second on a 2000 MIPS von Neumann processor.

As mentioned above, the ZISC036 has a built-in RCE-like [3] learning mechanism. Triggering this mechanism requires a simple user interface scheme, once the evaluation of the input vector has

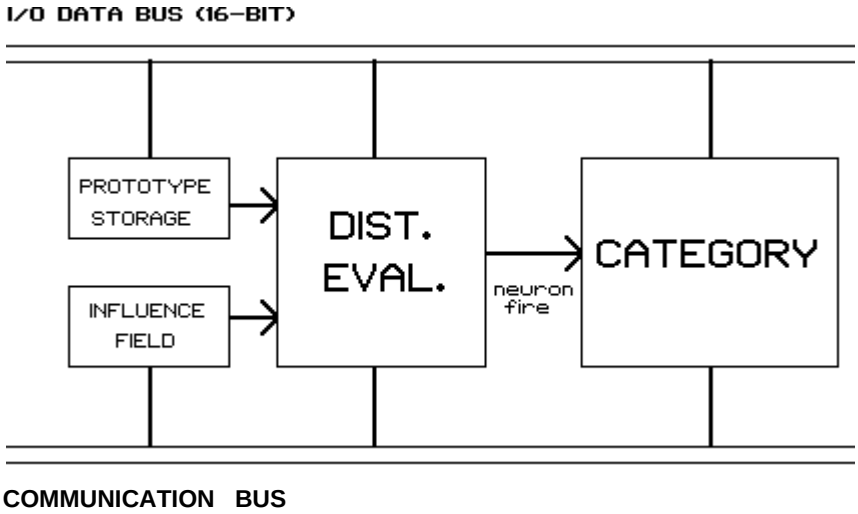

Figure 3. Block diagram of one of 36 neurons in the ZISC036 circuit.

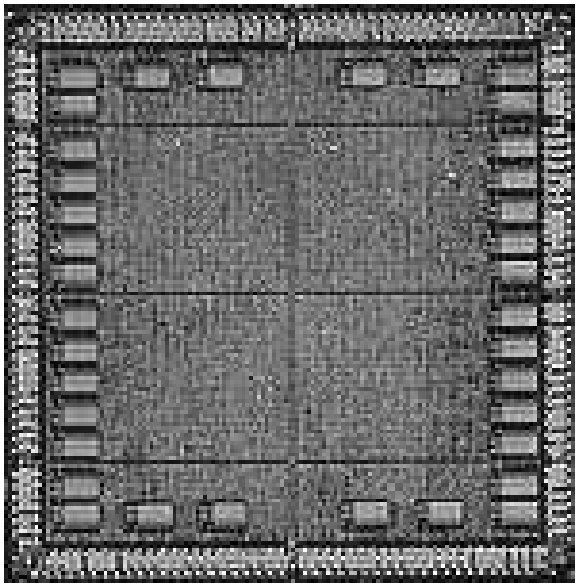

Figure 4 Die plot of the ZISC036. One can see the 36 register files for the neuron prototypes. The vertical line and the three horizontal lines are the power buses.

been completed. The learning of the chip can thus be refined at any time. The process is claimed to be completed in less than 1 μs (at 20 MHz operation) for the cases of new neuron commitments.

The chip has two registers holding the values of the minimum and the maximum influence fields associated with the prototypes (MIF and MAF, respectively). These values may both have values between 0 and 16383, of course with MIF < MAF. The CLEAR signal will set the values to default values of 2 and 4096, respectively. Recall that if an input vector falls outside (inside) the influence field(s) of any (one or more) prototype(s) associated with one category, it is not recognized (i.e. declared belonging to that category). However, if the input vector lies within the influence field of two or more prototypes associated with different categories, it is declared as recognized but not formally identified.

Other features of the neural network include a "K-nearest neighbours´ function", which returns the distances between a vector feature and stored prototypes in ascending order of distance and a "save/ restore function", which makes it possible to transfer information from e.g. one network to another.

In many RBF applications a large number (100 - 1000) of neurons is required. The fact that the ZISC circuit is cascadeable and most of its 144 pins can be directly interconnected will simplify the design and allow for "ZISC-towers" (see below). Multi-layer configurations can be accomplished by either connecting several chips or subsetting the network and time-multiplexing the inputs.

The ZISC036 chip supports asynchronous as well as synchronous protocols, the latter when a common clock can be shared with the controller. Examples of these modes are given in the user´s manual and includes the use of separate address and data buses as well as PCI-like multiplexing, single, multiple and burst transfer. The ZISC036 (cf. fig. 2 and 3) is conveniently regarded as a coprocessor device. As such a device, it must be controlled by a micro-controller or a state machine (accessing its registers). The input/output of the ZISC036 [8] is managed through (i) the chip select signal (CS_),

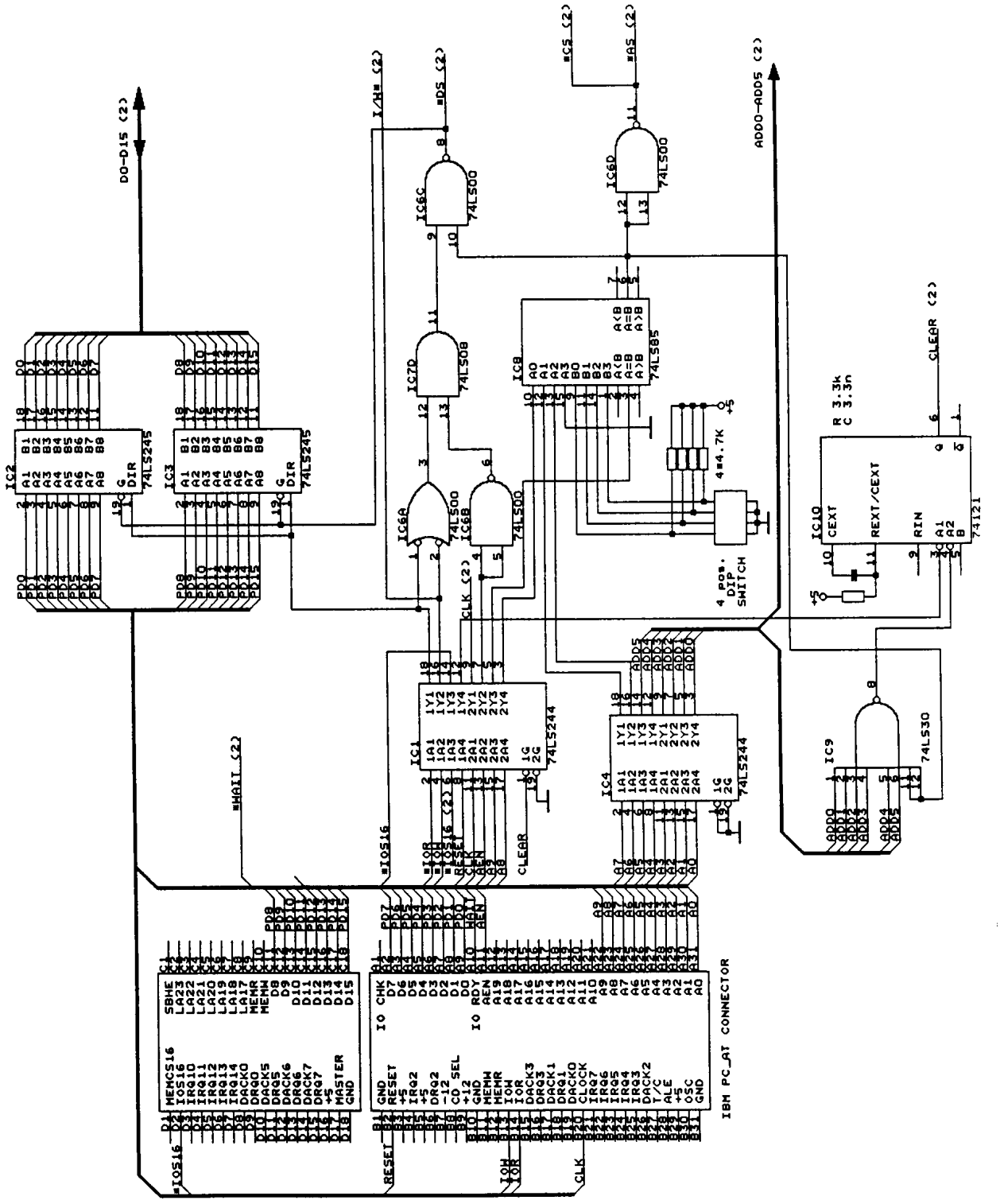

Figure 5 Schematics of the PC-ISA bus interface

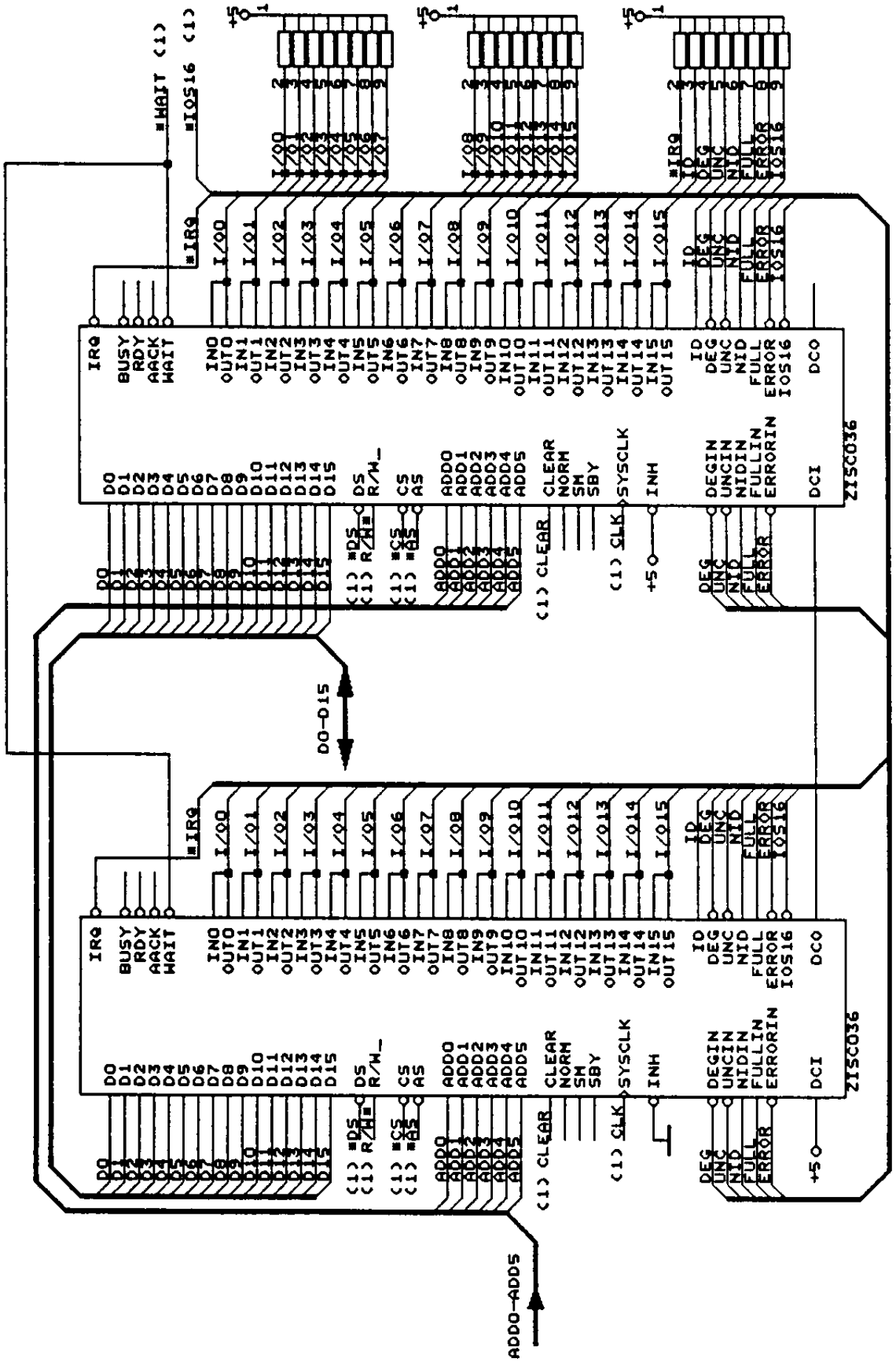

Fig. 6 Schematic of cascading the ZISC036

(ii) the 6-bit address bus and two control signals (A(5:0), AS_ and AACK_), and, (iii) the 16-bit data bus and five more control signals (D(15:0), DS_, RDY_, IOS16_, WAIT_ and RW_). A 4-bit decision bus, carrying the results of the classification, allows the use of the chip in stand alone mode. Interface examples for computer buses like ISA, MicroChannel, etc, as well as *i*80X86, MC6800 and PowerPC microprocessors, are found in the ZISC036 data book [8].

The ZISC036 chip is said [9] to be the first element of a neural network product family and designed as an entry product. It is implemented in standard cell IBM CMOS2 1.0 micron technology (cf. fig. 4) and has approximately 400 000 transistor functions [10]. It is encapsulated in a 144-pin 20 x 20 x 1.4 mm plastic flat pack for surface mounting.

4. The hardware implementation

We have chosen to make an initial implementation on a PC-486 computer running at 33 MHz and equipped with a standard ISA-bus. The time diagrams of ZISC036 and ISA-bus are very similar and the interface in asynchronous mode turns out to be quite simple. The major goal of the implementation is to gain experience of the chip itself and learn how to write software in high level language. A more "sophisticated" implementation is presently designed for the VME-bus [11].

The complete interface including two ZISC036 chips is shown in figs. 5 and 6. In the former drawing the ISA-bus is shown on the left hand side. The circuits labelled IC1 and IC4 (74LS244), are dual 4 bit line drivers. IC4 and one half of IC1 are always enabled and passes the bus signals they buffer straight through. One half of IC1 will lock during CLEAR signal only. It pass IOR, IOW and RESET signals from bus and IOS16 to bus. IC2 and IC3, the 74LS245, are bi-directional 8 bit bus transceivers. The direction of data flow is determined by the IOR signal. The function of IC8, the 74LS85, is to compare the levels arriving from two logical sources and output a high signal when they are equal. Address bit A9 is used as an enable gate to the 74LS85 and must be high to permit further arbitration. Address lines A8-A6 are compared with the inputs that come from the resistor bank and switch. In the present case, the board is designed to use the address field H300-H33F. The address H33F (IC9) is used as CLEAR signal.

ZISC036 chips are surface mounted on small piggyback PCB´s with two rows of single-line pin connectors along each edge (4 * 40 connectors[1)]). This makes it easier to work with the chips on the prototype card (connecting LogiScopes, etc), but also to allow for future cascading of ZISC chips. This cascading is carried out by using both male and female connectors, allowing the PCBs to be mounted on top of each other, i.e. the ZISC036 is ”cascaded” in a short daisy chain. Figure 6 shows that, in fact, all pins but two should be connected and this feature of the ZISC036 makes it possible to construct "ZISC-towers". However, in the present case, the two PCBs are mounted on the AT-card directly. It should be stressed that the interface bus is identical in a single or multi-device neural network. All input signals are applied in parallel to each chip. The address and bi-directional Data Bus are also connected in parallel to every device. These signals include *CS, *AS, A(5:0), R/W*, *DS, D(15:0), CLEAR, and SYSCLK.

[1)] The presently used prototypes have 160 pins, while the commercial ones (also used for the VME-board implementation) have 144 pins.

The DCO output of the first ZISC device must be connected to the DCI input of the second device. Furthermore, the DCI input of the first device must be connected to a "high level" (+5 V). The DCO output of the second chip does not require any similar connection. The inter-chip connection bus involves leads for input and output signals, which must be connected by pair, each pair to a pull-up resistor. The 21 pairs of input/output signals must be connected that way:

- input signals: IN(15:0), *DEGIN, *UNCIN, NIDIN, FULLIN, *ERRORIN
- output signals: OUT(15:0), *DEG, *UNC, NID, FULL, *ERROR.

The signal *INH of only one device, whatever it is, must be pulled up (connected to +5 V) while another device must have the input *INH connected to GND. The device that is tied up drives the bi-directional bus, and hence this one is referred to as the driving device.

5. ZISC PC-486 software

In this chapter we will describe some of the software developed to test the ZISC036. All software was written in Borland C (version 4.0) and based on information obtained from IBM Microelectronics in Paris [10]. As many neural nets of the present type are tested to recognize characters, we have also chosen to do so. Thus we start with a network in which a simple picture can be drawn by using "." for a picture element that is off, and "#" for a picture element that is on. A set of 8 * 8-byte patterns of the capital and lower-case letters (26+26) was used, and presented to the chip together with their pertinent ASCII codes. Hence, the parameters used during learning were as followes:

Number of components of input vectors: 64
Number of categories: 52
Norm used: L1, {i.e. L_1 in eq. (3)}

The only constraint on the picture is that all the input pictures must be the same size and dimension. The outputs of the net will be numbers as shown in the example below.

input	desired output	numerical input	desired numerical output
. . . . .		46 46 46 46 46 46 46 46	
. . . . # . . .		46 46 46 46 35 46 46 46	
. . . # . # . .		46 46 46 35 46 35 46 46	
. . # . . . # .		46 46 35 46 46 46 35 46	65
. # . . . . . #	**A**	46 35 46 46 46 46 46 35	
. # # # # # # #		46 35 35 35 35 35 35 35	
. # . . . . . #		46 35 46 46 46 46 46 35	
. # . . . . . #		46 35 46 46 46 46 46 35	

After supplying the 26 patterns of the capital letters and the 26 patterns of the lower-case letters together with ASCII-codes, the contents of the neurons were tested by putting the chip into SAVE/RESTORE mode and reading the contents of the first 52 neurons (having the context number 0; cf

the ZISC036 data book). It was found, that the NAIF (*N*euron *A*ctual *I*nfluence *F*ield) values strongly depend on the input component values. The larger difference between the two possible component values has given the larger NAIF value and vica versa.

This network was tested with several input data sets having different level of noise. The noise was introduced into the input pattern by changing one or more picture elements to its opposite value. Clearly we may define the 52 patterns using a 64 * 1 vector $\mathbf{v}_n^t = [0, 0, ... 1, ... 0, 1]$, where a 0 represents the background or "." and a "1" respresents the "#". We also define a vector $\mathbf{u}_n^t = [1, 1, ... 0, ... 1, 0]$,

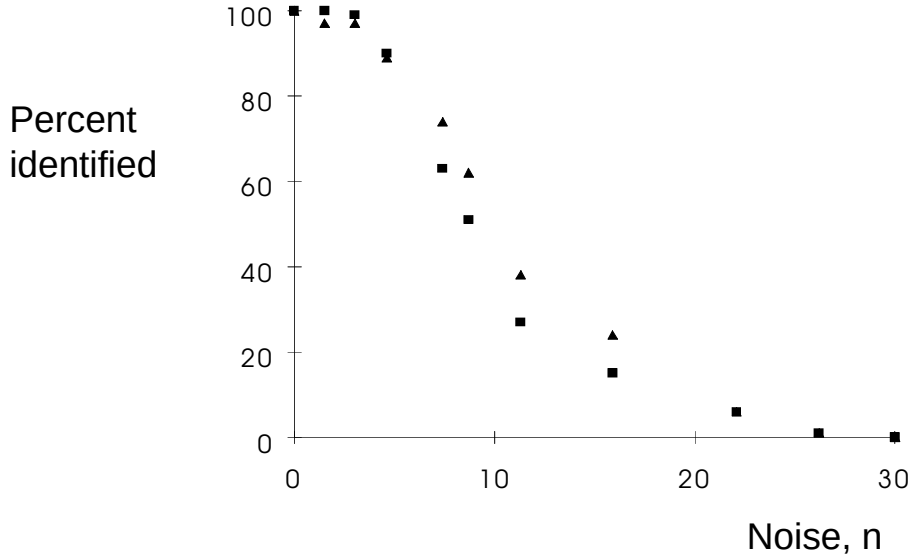

Figure 7. Result of testing the ZISC036 RBF-net discussed in the text. The result from the ZISC036 h/w run with 52 prototypes is shown by squares, while the triangles respresent a run using only 36 prototypes. Results (not included in fig) from software codes (DynaMind and Suzy) discussed in the text all yield values well below the points shown here.

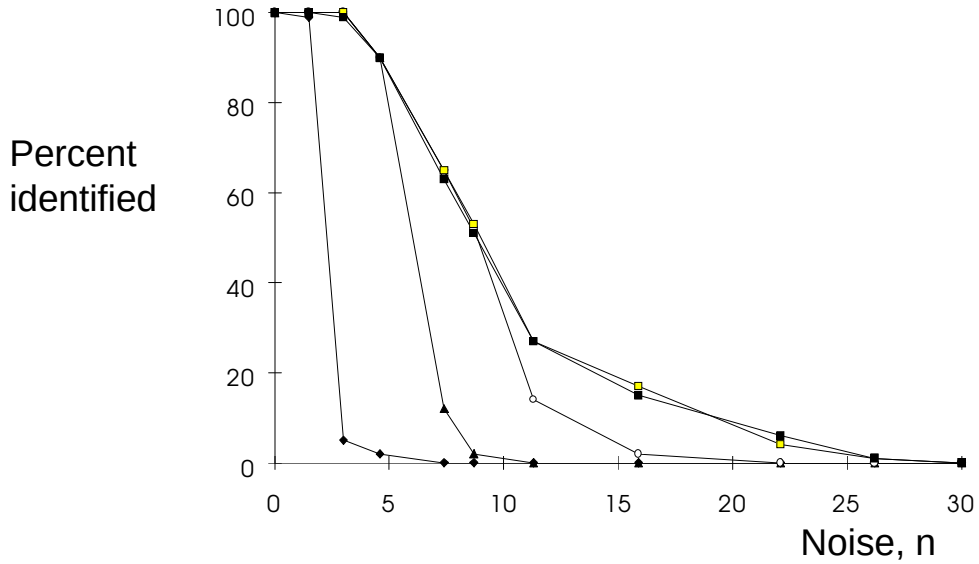

Figure 8 Same as fig 7 but changing the maximum influence field (MAF). Cases shown are for 36 prototypes and MAF = 200 (diamonds), 400 (triangles), 1000 (circles), 2048 (open squares), and 4096 (filled squares, same as curve in fig. 7). Values larger than MAF > 2048 all yield similar curves. The minimum influence field was kept constant in all cases and equal to the default value (2).

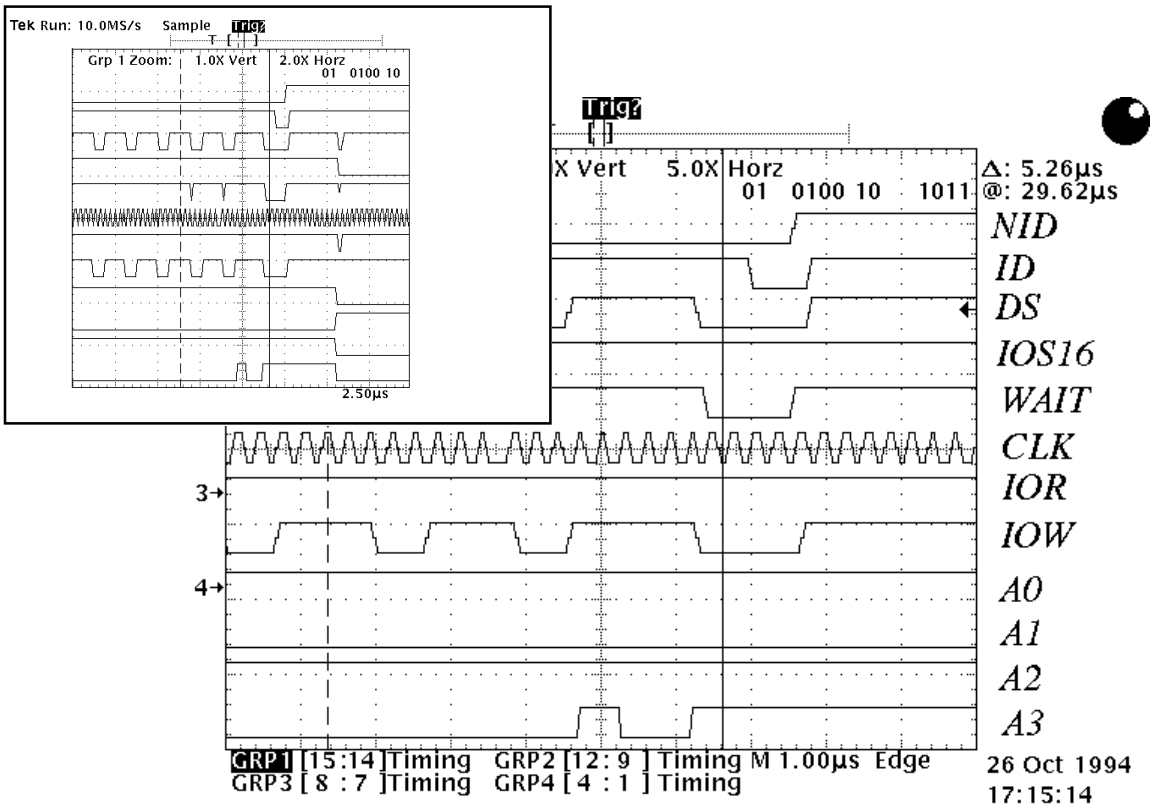

Figure 9. Timing diagrams of the present ZISC036 implementation recorded using a Tektronix TLS216 Logic Scope. The signal mnemonics of the ZISC036 are shown on the right hand side. In the large diagram the time axis has been expanded 2.5 times.

and could then require that for a test vector **t**, a "clean match" is achieved if the first scalar product ($\mathbf{v}_n^t\mathbf{t}$) equals the number of pixels in the pattern and if the second one ($\mathbf{u}_n^t\mathbf{t}$) is equal to zero. The noise in a test vector **t** is then defined as

$$n(t) = \min 0.5\,[\,1 - (\mathbf{v}_n^t * \mathbf{t})/n_{il} + (\mathbf{u}_n^t * \mathbf{t})/n_{i0})], \qquad (5)$$

where the minimum is searched for in all 52 templates, i.e. $0 < i < 53$, and the n_{il} and n_{i0} are the number of "1" and "0", respectively.

6. Performance

Figure 7 shows the results obtained using the code for the noisy character identification discussed above (squares). In this example, we have used the default values for the minimum and maximum influence field (cf. sect. 3). For comparison we have also included the results using only 36 prototypes (triangles). Runs with software codes like BrainMaker [12], Suzy [13] as well as running RBF in MatLab [14], all yield data points below the ones shown in fig 7. BrainMaker is a feedforward network package and was tested in several configurations. Suzy is a public domain RBF-type written in Borland C++ and Turbo Vision. It uses a non-linear step function and the delta-rule to find the weights of the output layer. Although both DynaMind and Suzy yields > 99.5% correctly identified

characters for zero noise, the performance drops quickly when some noise is added. In the case of Suzy, this could possibly be explained by too small maximum influence radii. However, this cannot be changed in the present version of the code. Hence, we can conclude that the ZISC036 is performing quite well. Training is fast and could very well serve as a coprocessor board in a PC/AT/486 to speed up extensive RBF calculations.

To check the effect of varying the maximum influence field (MAF), we performed a test resulting in fig. 8. This figure shows that lowering the MAF worsens the performance. Furthermore, for MAF > 2048 one sees no improvement performance of the chip. The curve for MAF = 200, however, is still better than that obtained using the code Suzy (cf. above). This supports the assumption that the maximum influence field may be too small in this code. Finally, it could be mentioned that, for most cases, the fraction of identified but not classified (i.e. multi-cathegory firings) input vectors is small or equal to zero. The exceptions are the cases with a few percent noise where the percentage of unclassified cases is around 10%.

We conclude this section by showing time diagrams (fig. 8) for the present PC-486 implementation. Among other things, one could conclude that the result of the classification is available one clock cycle after the last data has been presented to the chip.

7. Summary

The ZISC036 has some very interesting features, that should make it very useful for a variety of hardware applications. In particular, the easy way of cascading the chips due to the arrangement of the various buses (cf. fig. 6) is of importance. A "ZISC-tower" of 8 or 9 chips would yield the 300 neurons required in many applications. Assuming that further versions will have more neurons per chip, it is reasonable to expect similar towers with 1500 to 3000 neurons per tower.

Although the ZISC036 is a very "simple" classifier with a limited capability to generalize, it has the advantages of RBF networks: short training time and good confidence estimates. Greater generalization could be achieved by arranging several networks in parallel. Each network would be trained with the same data but with different approaches (parameters), and the results would then be fed to an "exiting" network trained to do the selection of the final result. Such an arrangement of networks is again made easy by the structure of the buses of the ZISC. However, sometimes a rather "rigid" neural network may be desirable, e.g. for monitoring deviations in patterns and finding novel data (beyond the "known" prototypes).

We also plan to use the ZISC chips in combination with other hardware that is more redundant and has a larger power to generalize [15]. This implementation would follow the suggestions of McClelland *et al.* [16] with respect to the learning process in the *hippocampus* and *neocortex* of the human brain [17]. The ZISC chips would correspond to the hippocampus and feed the output to the generalizing second network. This network may be a feedforward net trained by using standard backpropagation, simmulated annealing, Boltzman, etc algorithms or even a von Neumann computer running a network code with high redundance and capability to generalize.

Although RBF networks have been around for a while, there seems to be a renewed interest in using this type of nets for pattern recognition as well as control [18]. We are considering applications in high energy physics, where RBF-nets would serve as level 2 triggers, identifying regions of interests, separating electromagnetic and hadronic jet events, etc. The RBF-type of neural network may be appropriate for complex correlations between outputs from multi-detector systems. As is well known, and pointed out in e.g. ref . [19], some neural networks do not manage to separate the "double-spirals",

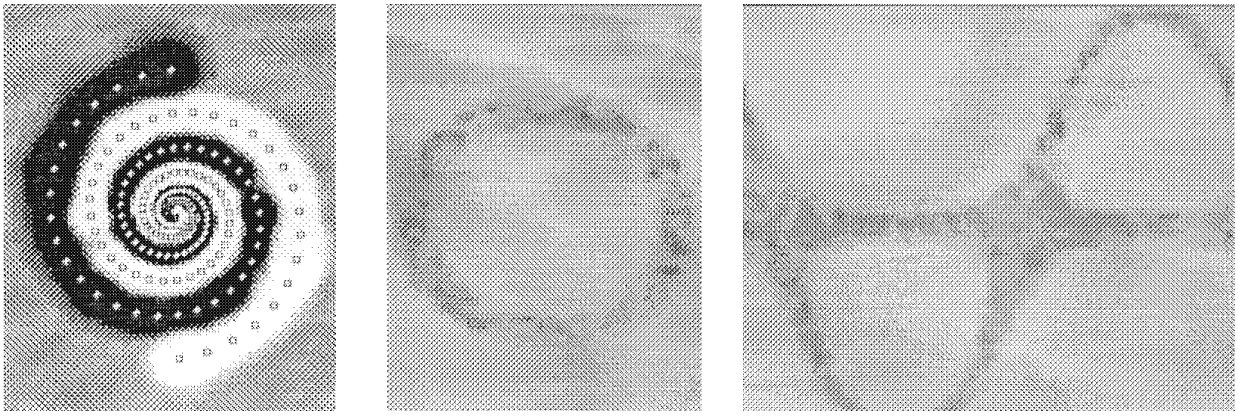

Figure 9 Some patterns suitable for RBF-type of neural networks? The famous double spiral (here after ref [19], cf. also ref. [22]) is shown (left). The middle and right hand figures are from high energy physics research [21] and show 2-particle invariant mass correlations (originally a six dimensional parameter space).

while RBF-like nets usually do quite well. The resemblence between such patterns (double spiral) and e.g. correlation patterns in high-energy physics [21] is suggestive for further investigations (cf. fig. 9). This is also the case for the concept "neural network as a measuring device" discussed in ref. [23]. As mentioned above, several of these applications may require RBF nets with different radial basis functions (c.f. eg. ref. [18] sect. 3.3.4) depending on the task as well as several RBF-type of nets in parallel. There are also some new training algorithms like the "Dynamic Decay Adjustment" or DDA [19-20], which may be very useful for the applications mentioned. Simple global scaling of the input variables have also been found to improve the performance [24] of RBF-nets.

Acknowledgements

The present work was carried out under contract with the Swedish Research Council for Engineering Sciences (TFR), which is gratefully acknowledged. We would also like to acknowledge the support and comments of Jean-Pierre LeBouquin at IBM Essones Components Lab., Corbeil Essones, France, and Guy Paillet of Neuroptics Consulting, Montpellier, France. Three of us (M.M., G.S., and G.S. would like to thank the Swedish Academy of Engineering Sciences (IVA), and the Royal Academy of Sciences (KVA) for fellowships.

References

[1] M.J.D. Powell, *Radial Basis functions for multivariable interpolation: A review*, IMA Conference on Algorithms for the Approximation of Functions and Data, RMCS, Shrivenham, UK, 1985 143 - 167

[2] J.E. Moody and C.J. Darken, *Fast learning in networks of locally-tuned processing units*, Neural Computation, 1, 1989, 281 - 294

[3] D.L. Reilly, L.N. Cooper and C. Elbaum, *A neural model for category learning.* Biological Cybernetics 45, 35-41 (1982)

[4] S. Renals, *Radial basis function for speech pattern classification*, Electronics Letters, 25, 1989,437 - 439

[5] S. Chen, C.F.N. Cowan, and P.M. Grant, *Orthogonal least squares learning algorithm for radial basis function networks*, IEEE Trans. on Neural Networks, vol. 2, no. 2, March 1991, 302-309

[6] Simon Haykin, Neural Networks, *A Comprehensive Foundation*, IEEE Press, ISBN 0-02-352761-7

[7] C. Park, K. Buckmann, J. Diamond, U. Santoni, Siang-Chun The, M. Holler, M. Clier, C.L. Scofield and L Nunez, *A Radial Basis Function Neural Network with On-chip Learning*, Intel Corp and Nestor Inc.

[8] *ZISC036 data book*, IBM Essones Components Development Laboratory, IBM Microelectronics, F-91105 Corbeil-Essonnes, France

[9] J-P LeBouquin, *IBM Microelectronics ZISC, Zero Instruction Set Computer, Preliminary Informa tion*, Poster Show WCNN, San Diego, CA, 1994 and addendum to conference proceedings

[10] J-P LeBouquin, IBM Essones Components Development Laboratory, F-91105 Corbeil-Essonnes, France, *priv. comm.*

[11] Th. Lindblad *et al*. to be published

[12] DynaMind User´s Manual, NeuroDynamx, Denver, CO

[13] Tom Grove, SUZY (ver. 1.0) 1993

[14] MatLab, Neural Network Toolbox (ver. 2.0) MathWorks Inc, Natick, MA

[15] Th. Lindblad, C.S. Lindsey, M. Minerskjöld, Å. Eide and G. Szekely, *Consolidated Artificial Neural Networks, to be publ.*

[16] J. L. McClelland, B. L. McNaughton and R. C. O´Reilly, *Why there are Complementary Learning Systems in the Hippocampus and Neocortex: Insight from the Successes and Failures of Connectionist Models of Learning and Memory*, Technical Report PDP.CNS.94.1, March 1994 CMU, Pittsburg, PA

[17] D.H. Freedman, *A Romance Blossoms Between Gray Matter and Silicon* , Science 265, Aug. 1994, pp 889-890

[18] M. Brown and C. Harris, *Neurofuzzy Adaptive Modelling and Control*, Prentice Hall, 1994

[19] M.B. Berthold and J. Diamond, *Boosting the Performance of RBF Networks with Dynamic Decay Adjustment*, Dortmund Fuzzy Days, Adv. Neural Information Processing. Eds G. Tesauro, G. Touretsky and. J. Alspector, Morgan Kaufmann, Calif.

[20] M. Berthold, *A Time Delay Basis Function for Phoneme Recognition*, Proc. of the IEEE International Conference on Neural Networks, 1994

[21] R.K. Bock, J. Carter and I.C. Legrand, *What can artificial neural networks do for the global second level trigger*, Atlas/DAQ-No-11, EAST 94-08 (March 22, 1994), CERN, Geneva

[22] S, Fahlman and M. White, *The Carnegie Mellon University Collection of Neural Net Benchmarks*, ftp from cs.cmu.edu in /afs/cs/project/connect/bench

[23] Å. Eide and Th. Lindblad, *Neural Networks as Measuring Devices*, Nucl. Instr. Meth. A317(1992) 607-608

[24] S. M. Botros and C. G. Atkeson, *Generalization Properties of Radial Basis Functions*, Advances In Neural Information Processing Systems, I-707-713, Ed. D. Touretzky, Morgan Kaufmann Publishers, CA, ISBN 1-55860-015-9