A Fuzzy Clustering Neural Networks (FCNs) System Design Methodology

David Zhang and Sankar K. Pal

***Abstract*—A system design methodology for fuzzy clustering neural networks (FCNs) is presented. This methodology emphasizes coordination between FCN model definition, architectural description, and systolic implementation. Two mapping strategies both from FCN model to system architecture and from the given architecture to systolic arrays are described. The effectiveness of the methodology is illustrated by: 1) applying the design to an effective FCN model; 2) developing the corresponding parallel architecture with special feedforward and feedback paths; and 3) building the systolic array (SA) suitable for very large scale integration (VLSI) implementation.**

***Index Terms*—Neuro-fuzzy clustering, systolic array, very large scale integration (VLSI).**

I. Introduction

THERE have been a number of approaches for designing fuzzy clustering neural networks (FCNs) have been considered. Simpson [1] discussed on fuzzy min-max neural networks in fuzzy clustering. Pal *et al.* [2] developed a fuzzy clustering network based on the Kohonen network. Mitra and Pal [3], [4] described a self-organizing neural network, which is capable of handling fuzzy input and of providing fuzzy classification. All these approaches are concerned with algorithms; their behaviors and characteristics are primarily investigated by simulation on general-purpose computers. The fundamental drawback of such simulators is that the spatiotemporal parallelism inherent in the processing of information using neural networks is lost entirely or partly. Moreover the computing time of the simulated network, especially for large associations of nodes tailored to application-relevant tasks, grows to such orders of magnitude that a speedy acquisition of neural "know-how" is hindered or made impossible. This makes the actual fuzzy clustering applications difficult to implement in real time. Therefore, it is essential to implement FCN in a very large scale integration (VLSI) medium.

However, it cannot be assumed that FCN models developed in computational neuroscience, at a high level, are directly implementable in silicon. This is because the technology, the physical devices and the circuits severely limit the performance of integrated FCN. Systolic array (SA) can offer flexibility, programmability, and precision in computation, coupled with the advantages of large pipelined throughput and local interconnections in VLSI implementation [5]–[7]. In the present article, we describe how an efficient FCN system can be realized by considering the special features and aspects of the fuzzy clustering technology, designing their special-purpose architectures, and mapping the neural networks onto the corresponding systolic arrays (see Fig. 1).

Manuscript received May 24, 1999; revised March 21, 2000 and June 22, 2000. The work was supported in part by the UGC/CRC, Hong Kong Government and the central fund, Hong Kong Polytechnic University.

D. Zhang is with the Department of Computing, Hong Kong Polytechnic University, Kowloon, Hong Kong.

S. K. Pal is with the Machine Intelligence Unit, Indian Statistical Institute, Calcutta 700035, India.

Publisher Item Identifier S 1045-9227(00)07867-X.

II. FCN Model

A basic FCN model using clustering competitive network is illustrated in Fig. 2. Each node represents a fuzzy cluster and the connecting weights from the inputs to a node represent the exemplar of that fuzzy cluster. The square of the Euclidean distance between the input pattern and the exemplar is passed through a Gaussian nonlinearity. The output of the node, therefore, represents the closeness of the input pattern to the exemplar. The degree of possibility that each input pattern belongs to different fuzzy clusters is calculated in the final membership level.

A fuzzy cluster criterion, called quality of fit or Q, is defined as the sum of all output values of the nodes over all input patterns. That is $Q = \sum_{k=1}^{P} Q_k = \sum_{k=1}^{P} \sum_{i=1}^{M} C_{ki}$, where P and M are the respective numbers of input patterns and nodes, and C_{ki} is the output of node i when the input pattern is $\mathbf{x}_k = (x_{k1}, x_{k2}, \ldots, x_{kN})$, $C_{ki} = \exp(-((d_{ki})^2/2\sigma^2))$. The weight vector connecting the inputs to node i is $\mathbf{w}_i = (w_{i1}, w_{i2}, \ldots, w_{iN})$. The Euclidean distance between $\mathbf{w}_i$ and $\mathbf{x}_k$ is defined as $(d_{ki})^2 = \sum_{j=1}^{N} (x_{kj} - w_{ij})^2$. The weight vectors, $\mathbf{w}_i$ $(i = 1, 2, \ldots, M)$, can also be viewed as the parameters of the Gaussian functions that determine their locations in the input space. Since the fuzzy clusters are high concentrations of the input patterns in the input space, locating the weight vectors at or close to the centers of these concentrations will insure a maximum for the quality of fit criterion.

This is clearly an optimization problem where the objective function is the quality of fit and the variables are the coordinates of the centers of the Gaussian functions, i.e., the weight vectors. The change in the weight on the objective function is $\Delta w_{ij} = \eta(\partial Q/\partial w_{ij}) = \eta(\partial/\partial w_{ij})\{\sum_{k=1}^{P} \sum_{i=1}^{M} C_{ki}\}$. This is equal to $\Delta w_{ij} = \eta \sum_{k=1}^{P} \partial C_{ki}/\partial w_{ij}$, where the second summation, $\sum_{i=1}^{M}$, was dropped since w_{ij} appears only in one term. Note that $\partial C_{ki}/\partial w_{ij} = (\partial C_{ki}/\partial d_{ki})(\partial d_{ki}/\partial w_{ij}), \partial C_{ki}/\partial d_{ki} = (-1/2\sigma_i^2) C_{ki}$ and $\partial d_{ki}/\partial w_{ij} = -2(x_{kj} - w_{ij})$. This means $\Delta w_{ij} = \eta \sum_{k=1}^{P} (1/\sigma^2) C_{ki}(x_{kj} - w_{ij})$, where η is a constant of proportionality and σ^2 is the variance of the function of the node. It is clear that this formulation utilizes local information available at the weight itself and the node it is connected to. There is no need for an external orientation subsystem to decide which weights are to be increased since all the weights are adapted after accumulating the errors over all input patterns. The amount of change in the weight is a function of the distance between the input pattern and the weight vector.

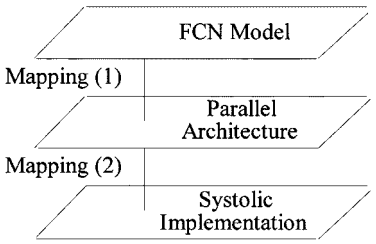

Fig. 1. A system design methodology for FCN implementation.

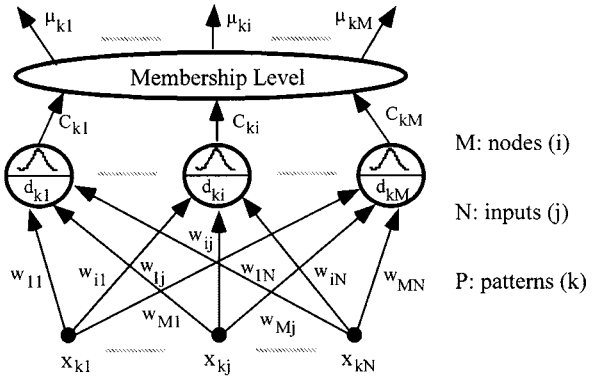

Fig. 2. Fuzzy clustering neural network (FCN) model.

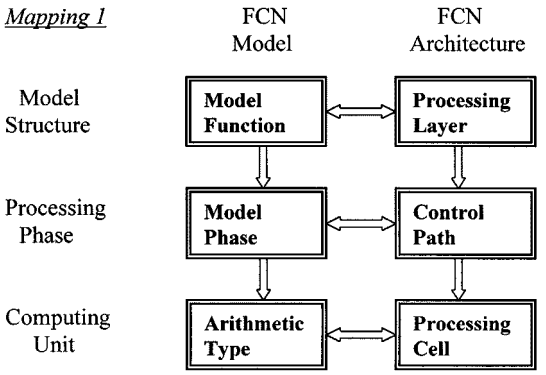

Fig. 3. Mapping strategies from FCN model to architecture.

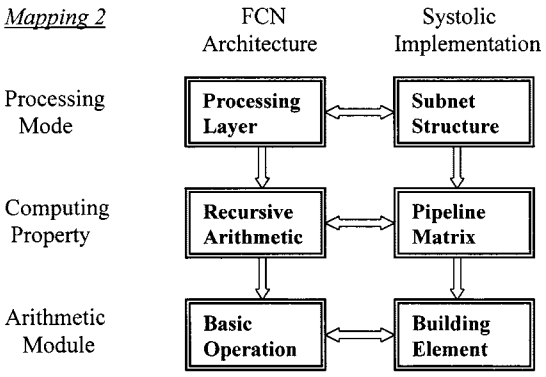

Fig. 4. Mapping strategies from FCN architecture to systolic implementation.

To introduce a fuzzy competition mechanism between the nodes, a membership function for fuzzy clustering is required. In other words, a partition of the input pattern space, $X = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_P\} \subset \Re^N$, into fuzzy clusters, Z_i $(i = 1, 2, \ldots, M)$, is associated with the membership functions $\mu_{Z_i}: X \rightarrow [0, 1]$. Assignment of input patterns to different clusters can be given in terms of a fuzzy cluster membership matrix $U = [\mu_{ki}]$, where the element μ_{ki} denotes the degree of belonging of the input pattern k to the fuzzy cluster $i, \mu_{ki} = \mu_{Z_i}(\mathbf{x}_k) = C_{ki}/Q_k$. It is evident that the elements of U are subject to $P > \sum_{k=1}^{P} \mu_{ki} > 0$ and $\sum_{i=1}^{M} \mu_{ki} = 1$, where $1 \leq i \leq M$ and $1 \leq k \leq P$. Using the fuzzy cluster membership element μ_{ki} to participate in the corresponding weight change, a fuzzy competitive learning update rule which moves the weight vectors toward their respective fuzzy cluster centers can be represented as $\Delta w_{ij} = \eta \sum_{k=1}^{P} (1/\sigma^2) C_{ki}(x_{kj} - w_{ij}) \mu_{ki}$. We can obtain a variation of this learning algorithm by letting $\Delta_k w_{ij} = \eta(\partial Q_k / \partial w_{ij})$. Note that the direction of the gradient only guarantees a locally increasing direction. To avoid instability of the algorithm, the step taken in the direction is usually chosen to be very small by the control parameter, η. Thus, if η is sufficiently small, $\Delta w_{ij} \approx \sum_{k=1}^{P} \Delta_k w_{ij}$. This means the change in weights will be approximately equal to Δw_{ij} if the weights are updated after the error is computed corresponding to an input pattern, unlike the batch mode learning. Considering the approximation, we can obtain $\Delta_k w_{ij} = \eta(1/\sigma^2) C_{ki}(x_{kj} - w_{ij}) \mu_{ki}$.

III. Parallel Architecture

A. Mapping Strategies

An FCN architecture is specified by its network topology and node characteristics. The network topology defines how each node is connected to other nodes. The node characteristics define the function which combines the various inputs and weights into a single quantity as well as the function that then maps this value to an output. Considering parallel network topology, some mapping strategies from an FCN model to the architecture are defined as follows (See Fig. 3):

1) Model Structure Mapping: In the FCN model, each function, like competitive and membership function, is mapped as an independent processing layer and their connection patterns within and between layers are defined.

2) Processing Phase Mapping: Often two processing phases, searching and learning, are given in the FCN model. They are able to be performed in architecture design by special control paths, i.e., feedforward and feedback path, respectively.

3) Computing Unit Mapping: Depending on different functional arithmetic types in each processing layer, such as weight computing and pattern summing, their processing cells can be built to achieve the given input–output functions.

B. FCN Architecture: Processing Cells

The FCN architecture comprises three kinds of processing cells, including weight, node, and output, plus adder (Σ). We

have embedded all functions from the FCN model in this architecture so that on-line learning and parallel implementation are feasible.

Both node cell and adder (Σ) can be achieved by many current approaches [5], [8]. The output processing cell is used to obtain both the membership element, μ_{ki}, and the partial product for the fuzzy competitive learning update rule (Section II). The two inputs, C_{ki} and Q_k, come from the outputs of node i and the adder (Σ) when the input pattern is $\mathbf{x}_k$. The output of the partial product is $S_{ki} = AC_{ki}\mu_{ki}$, where A is a control parameter for the operator. The output cell can be built using multiplier and divider operators. A weight cell is used to store and change weight value as well as implement the related arithmetic. It is mainly composed of four different memory elements, i.e., accumulation memory $(\Sigma\Delta_k w_{ij})$, weight memory (w_{ij}), difference memory (g_{ijk}) and its square memory (g_{ijk}^2), and five operators (two adders, one subtractor, one multiplier and one unit that generates square of number). In the feedforward processing, an input, x_{kj}, is given and the outputs of the cell are represented as $g_{ijk}^2 = (x_{kj} - w_{ij})^2$. In the feedback processing, the input S_{ki} is to come from the output cell and its corresponding arithmetic to implement the update rule is $\Delta_k w_{ij} = BS_{ki}g_{ijk}$, where B is the control parameter of the multiplier in the cell, and g_{ijk} is obtained in the previous processing and stored in the difference memory (g_{ijk}). We use the update rule in batch mode: $\Delta w_{ij} = \sum_{k=1}^{P} \Delta_k w_{ij}$. In this way, we can implement the Euclidean distance between weight vector $\mathbf{w}_i$ and input pattern $\mathbf{x}_k$ in the FCN architecture rewritten as $(d_{ki})^2 = \sum_{j=1}^{N} g_{ijk}^2$, where g_{ijk}^2 is as an input from the weight cell, w_{ij}, to the node cell i. The weights in the FCN architecture are changed in terms of the following functions: $w_{ij}^t = w_{ij}^{t-1} + \Delta w_{ij}$ and $\Delta w_{ij} = B\sum_{k=1}^{P} S_{ki}g_{ijk} = B\sum_{k=1}^{P} AC_{ki}\mu_{ki}g_{ijk}$, where w_{ij}^t and w_{ij}^{t-1} are the weights at time t and time $t-1$, respectively.

C. Performance Analysis

Based on the FCN architecture, hardware complexity can be represented as $H_{\text{FCNN}} = M(NH_{\text{weight}} + H_{\text{output}} + H_{\text{node}}) + H_{\Sigma}$, where N and M are the dimensions of the input and output spaces, respectively; H_{Σ} is the complexity of the adder (Σ); $H_{\text{weight}}, H_{\text{output}}$, and H_{node} are the complexities of three different cells, respectively. Note that H_{FCN} is independent of P, the number of the input patterns, and H_{Σ} has a linear complexity in the number of connections of the nodes, M. Therefore the attached cost of direct competition in the FCN architecture, H_{Σ}, can be compared with the other competitive architectures, such as the MAXNET [9] with the connective complexity, $M(M-1)/2$. This means that the connective cost of direct competition in the FCN is reduced by a factor of $(M-1)/2$.

In order to analyze the effectiveness of the FCN architecture, we take the architecture of fuzzy c-means (FCM) as our comparative target. The objective function for the FCM is given by [10], $J = (1/2)\sum_{i=1}^{M}\sum_{k=1}^{P}(\mu_{ki})^{\beta}(d_{ki})^2$, where its membership function is $\mu_{ki} = (1/\sum_{l=1}^{M}(d_{ki}/d_{kl})^{2/(\beta-1)})$. Thus, the fuzzy competitive learning update rule can be obtained as $\Delta_k w_{ij} = \eta\gamma_{\beta}(x_{kj} - w_{ij})$, where $\gamma_{\beta} = (\mu_{ki})^{\beta}[1 - \beta(\beta-1)^{-1}(1-\mu_{ki})]$, with $\beta \in [1, \infty)$. It has shown that these two architectures have identical structure and the same number of building elements, but they do differ in the complexities of the three kinds of cells. The node cell for the FCM architecture is characterized by the Euclidean distance $(d_{ki})^2$ rather than the Gaussian nonlinearity C_{ki}. The function characterizing each output cell is given by $\mu_{ki} = (1/(d_{ki}/D_k)^{2/(\beta-1)})$, where $(D_k)^2$ is the output of the adder (Σ) and is defined as $(D_k)^2 = \sum_{i=1}^{M}(d_{ki})^2$. The definitions of the cells in the two architectures, discussed above, are summarized in Table I. Here the number of the elements required is indicated in parentheses.

TABLE I
COMPARISON OF THE PROCESSING ELEMENTS BETWEEN FCN ARCHITECTURE AND FCM ARCHITECTURE

	FCN	FCM
Weight Cell *(M × N)*	$g_{ijk}^2 = (x_{kj} - w_{ij})^2$ $\Delta w_{ij} = \eta\Sigma S_i g_{ijk}$	$g_{ijk}^2 = (x_{kj} - w_{ij})^2$ $\Delta w_{ij} = \eta\sum(\mu_{ki})^{\beta}[1 - \beta(\beta-1)^{-1}(1-\mu_{ki})]g_{ijk}$
Node Cell *(M)*	$C_{ki} = \exp\left\{-\sum_{j=1}^{N} g_{ijk}^2 / 2\sigma^2\right\}$	$(d_{ki})^2 = \sum_{j=1}^{N} g_{ijk}^2$
Output Cell *(M)*	$\mu_{ki} = \frac{C_{ki}}{Q_k}$ $S_{ki} = \frac{1}{\sigma^2}C_{ki}\mu_{ki}$	$\mu_{ki} = \frac{1}{(\frac{d_{ki}}{D_k})^{2/(\beta-1)}}$
Adder (1)	$Q_k = \sum_{i=1}^{M} C_{ki}$	$(D_k)^2 = \sum_{i=1}^{M}(d_{ki})^2$

IV. Systolic Array Design

A. Mapping Strategies

It is clear that the complexity of the FCN stems not from the complexity of its nodes, but from the multitude of ways in which a large collection of these nodes can interact. Therefore, an important task is to build highly parallel, regular, and modular SA's that are attractive for VLSI techniques. Mapping from the FCN architecture to SA implementation can be achieved as follows.

1) Processing Mode Mapping: Here we partition a fuzzy clustering neural network into some basic subnets, each capable of performing an independent function. Often a subnet represents a layer in the neural networks. The subnets are implemented by a corresponding SA, which are then cascaded according to the architectural definition.

2) Computing Property Mapping: Each basic subnet function is reduced to a recursive form which is implemented by the corresponding pipeline matrix in terms of the systolic rules. In practice, this mapping transforms spatial parallelism to temporal parallelism.

3) Arithmetic Module Mapping: A basic operation in recursive arithmetic is implemented by a building element. A node can be divided into two parts: forming a weighted sum of N inputs and passing the result through a nonlinearity. The weighted sum can easily be integrated by a two-dimensional (2-D) recursive matrix using weight processing elements. To form the nonlinearity, a special element is defined which may be cascaded

with the recursive matrix as a bound node of its output. Mapping strategies from the FCN architecture to systolic implementation are shown in Fig 4.

B. FCN Systolic Arrays

Based on the given mapping strategies, the FCN architecture can be systematically implemented by the corresponding SA's, where two kinds of data flow paths, viz., feedforward and feedback exist. Each processing layer in the FCN architecture is achieved by the different SA's. Since two simple one-dimensional (1-D) SA's, output SA and node SA, can be easily built, we will concentrate here only on the discussion of 2-D SA.

Arranging properly the input data flow, X, the corresponding output of the nodes, C_{ki}, can be obtained by a feedforward SA with $M \times N$ weight PE's and M node PE's. The other 2-D SA is to implement the triangle SA with M adder PE's and $M[M-1]/2$ shifting registers, where an adder PE is defined as $S = r + s$ and $r' = r$. When the data flow $\{C_{ki}\}$ enters this array, each adder can accumulate its corresponding ΣC_{ki}. Note that after M steps both Q_k and $\{C_{ki}\}, 1 \le i \le M, 1 \le k \le P$, are available at the same time.

V. Conclusion

We have presented a novel system design methodology for FCN. This methodology offers flexibility, programmability, and precision in computation, coupled with the advantages of large pipelined throughput and local interconnections. Two mapping strategies from FCN model to architecture and from architecture to SA implementation are described. The effectiveness of the methodology is illustrated by applying the design to an effective FCN model, developing the corresponding parallel architecture, comparing with FCM architecture, and building the SA's suitable for VLSI implementation.

References

[1] P. Simpson, "Fuzzy min-max neural networks—Part 2: Clustering," *IEEE Trans. Fuzzy Syst.*, vol. 1, pp. 32–45, 1993.

[2] N. R. Pal, J. C. Bezdek, and E. C. Tsao, "Generalized clustering networks and Kohonen's self-organizing scheme," *IEEE Trans. Neural Networks*, vol. 4, pp. 549–557, 1993.

[3] S. Mitra and S. K. Pal, "Self-organizing neural network as a fuzzy classifier," *IEEE Trans. Syst., Man, Cybern.*, vol. 24, pp. 385–389, 1994.

[4] S. K. Pal and S. Mitra, *Neuro-Fuzzy Pattern Recognition: Methods in Soft Computing*. New York: Wiley, 1999.

[5] G. A. Jullien, W. C. Miller, R. Grondin, L. Del Pup, and D. Zhang, "Dynamic computational blocks for bit-level systolic arrays," *IEEE J. Solid-State Circuits*, vol. 29, pp. 14–22, 1994.

[6] J. Chung, H. Yoon, and S. R. Maeng, "A systolic array exploiting the inherent parallelisms of artificial neural networks," *Microprocessing Microprogram.*, vol. 33, no. 3, pp. 145–159, 1992.

[7] D. Zhang, *Parallel VLSI Neural System Designs*. New York: Springer-Verlag, 1998.

[8] D. Zhang, G. A. Jullien, and W. C. Miller, "A neural-like network approach to finite ring computations," *IEEE Trans. Circuits Syst.*, vol. 37, pp. 1048–1052, 1990.

[9] R. P. Lipmann, "An introduction to computing with neural nets," *IEEE ASSP Mag.*, vol. 4, pp. 4–22, 1987.

[10] J. C. Bezdek, "A convergence theorem for the fuzzy ISODATA clustering algorithms," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 2, pp. 1–8, 1980.