CFART: A New Multi-Resolutional Adaptive Resonance System for Object Recognition[*]

Hong-Yuan Mark Liao[††], Hai-Lung Hung[†], Chwen-Jye Sze[‡],

Shing-Jong Lin[‡], Wei-Chung Lin[†] and Kuo-Chin Fan[‡]

† Department of Electrical Engineering and Computer Science,

Northwestern University, IL 60208

‡ Institute of Computer Science and Information Engineering,

National Central University, Chung-Li, Taiwan

†† Institute of Information Science, Academia Sinica, Taiwan

Email: liao@iis.sinica.edu.tw

Tel:+886(2)788-3799 ext. 1811, Fax:+886(2)782-4814

Keywords: neural network, unsupervised learning, self organization, multi-resolutional representation

⋆ A preliminary version of this technical report will be presented at the **International Conference of SPIE on Visual Communications and Image Processing**, Orlando, Florida, March, 1996.

[*]This work was supported by the National Science Council of Taiwan under grant no. NSC 85-2213-E-001-017.

Abstract

In this report, we propose a cascade fuzzy ART (CFART) neural network which can function as an extensible database in a model-based object recognition system. The proposed CFART network contains multiple layers. It preserves the prominent characteristics of a fuzzy ART network and extends fuzzy ART's capability toward hierarchical representation of input patterns. The learning process of the proposed network is unsupervised and self-organizing, and includes a top-down searching process and a bottom-up learning process. The top-down and bottom-up learning processes interact with each other in a closely coupled manner. Basically, the top-down searching guides the bottom-up learning whereas the bottom-up learning influences the top-down searching by changing its searching fuzzy boundary. In addition, a global searching tree is built to speed up the learning and recognition processes. The proposed CFART can accept both binary and analog inputs. With fast learning and categorization capabilities, the proposed network is able to function like an extensible database and to provide an efficient multi-resolutional representation capability for 3D objects. Experimental results, using both synthetic and real 3D data, prove that the proposed method is indeed an efficient and powerful representation scheme.

1 Introduction

Three dimensional (3D) object recognition is the process of matching an object to a scene description to determine the object's identity and/or its pose (position and orientation) in space [1–4]. Any existing system capable of recognizing its input image must in some sense be model-based. In order to build a 3D object recognition system, one has to deal with two closely related subproblems [5,6] - that of model building and that of recognition. Basically, the way a model database is built will strongly affect the procedure used for recognition. In almost all existing model-based object recognition systems, the size of the database is fixed. That is, in the model building stage, the user has to store all the required models into the database. In the recognition stage, the system has to generate a description for an unknown input object and then try to match it with different object models located in the database. If a model is found to have a high degree of similarity with an unknown input object, then the unknown object will be assigned to a corresponding category. Otherwise, the system simply returns a response to report that no object model can match the current input. In general, this kind of system is sufficient for an automation line. However, if one needs to build a more flexible system which functions like a human being, then an extensible database is indispensable. An extensible database means that whenever the system encounters an unknown input that does not belong to any category in the database, the database will automatically include it as a new model. In order to build such a database, one should consider the following requirements: (1) self-organization, i.e., the training data set should automatically partition the feature space based on each sample pattern's characteristics; (2) extensible, i.e., if a new object is input, the system creates a new space in the feature space for the new category.

The above mentioned requirements basically match the characteristics of an unsupervised-learning-based neural network, such as the ART-series neural networks [7–11]. The unsupervised learning problem can be stated as being that of identifying the classes in an input set of patterns. Unsupervised learning neural networks, therefore, are capable of grouping similar patterns into classes automatically. Upon presentation of a given input pattern, the closest node in the network is activated to learn (i.e., modify its connection weights toward the current input vector). This process is referred to as a winner-take-all competition. In this way, each output node in the network is able to self-organize and can be used to represent a category of similar patterns. However, a serious problem associated with this scheme is that the number of categories the network can allocate for a given training set is restricted by the number of output nodes. An example reflecting this problem is the Self-Organizing Feature Map [12]. The Adaptive Resonance Theory (ART) neural network [7, 8, 11], on the other hand, can create a new node in response to new data during the training stage as well as during the recalling stage. A series of ART-type unsupervised learning networks [5, 13–17] has been developed and successfully applied to many applications. Some salient characteristics of ART-type networks are: (1) they can self-organize data set on line; (2) they can create new output nodes (categories) incrementally; and (3) they do not suffer from the problem of forgetting previously learned categories if the environment changes.

With full understanding of the characteristics of the required unsupervised neural network model, our problem at hand now is twofold. Since the input data (or patterns) of a 3D object recognition system are usually very complex by nature, the first problem we have to face is how to represent an unknown pattern in an efficient way. Then, the second problem will be how to design an appropriate neural network structure to correctly represent the above mentioned 3D object. Under the circumstances, if input data contains structural relationships, a single output layer of a neural network is definitely not enough to represent it. In general, people used to adopt a hierarchical structure [18–29] to represent a complex object or event. In order to design a more flexible neural network structure and to make it possible to represent a complex object, we extend the fuzzy ART network [11] to a cascade fuzzy ART (CFART) network [30]. A CFART network is a multi-layered neural network with an efficient top-down pattern matching scheme. Each module in a layer will learn the input data by either updating the weights of a matched category or creating a new category if all existing categories cannot represent it. Through the connections of categories in consecutive layers, the class hierarchy of input data is formed to construct a global searching tree from bottom to top (fine to coarse). The physical meaning of the hierarchical relationship is that categories in higher layers represent a more general view than do those in lower layers. For example, in a 3D model-based object recognition system, an object is usually described by multiple features. Using a CFART network as an extensible database, categories that have more specific descriptions will be stored in lower layers whereas categories having more general attributes will be stored in higher layers. Based on this kind of arrangement, a fine-to-coarse hierarchical object representation scheme with an efficient coarse-to-fine, top-down recognition capability is generated.

The organization of the rest of this report is as follows. In Section 2, the architecture and learning strategies of a CFART network are presented. In Section 3, a numerical example and a superquadric-based representation example are presented to demonstrate the capability of the proposed CFART network. In Section 4, discussion is given. Finally, in Section 5, concluding remarks are given.

2 Dynamics of a cascade fuzzy ART network

Fig. 1 shows the architecture and data flow of the proposed CFART network. A CFART network contains multiple layers. The module unit (MU) in every layer includes three parts, i.e., the top-down searching (TS) process, bottom-up learning (BL) process, and category database (CD). A global searching tree is constructed by linking those categories which have hierarchical relationships among different layers. At the beginning, the CD of every layer in the network is empty. Through the learning of an item of input data, the network creates or updates the corresponding weights of a category in the CD of every layer and builds a hierarchical linking path in a global searching tree. Given an item of input data, the network first performs a top-down searching process to locate a possible matching path in the searching tree. This searching process starts from the top layer and extends down to the bottom layer. The purpose of the searching process is to locate a path in the searching tree which links an appropriate category represented in a hierarchical manner in the database. Afterwards, a bottom-up learning process is triggered from the bottom layer and extends up to the top layer in order to create or update weights through the located path. In what follows, we shall discuss these processes in details.

2.1 Bottom-up learning

In the proposed approach, the top-down searching process is performed first to guide the bottom-up learning process. However, in order to understand the learning mechanism of a CFART network more clearly, the bottom-up learning process will be introduced first. In a CFART network, the bottom-up learning process is divided into two subprocesses, i.e., a "within-layer" learning subprocess and a "between-layer" learning subprocess. These two subprocesses are performed at the same time. In a learning module, MU_l, pattern matching and category selection are performed in TS_l (Fig. 1). After the winning category index, J^l, is generated from TS_l, its corresponding category in CD_l is then activated. This activated category is then trained by the input data through the within-layer learning subprocess. Let W_J^l be the weight vector of a winning category located in layer l and let I^l be the input data. The winning weight vector is updated by

$$W_J^{(new)} = W_J^{(old)} + \gamma(I^l \wedge W_J^{(old)} - W_J^{(old)}), \qquad (1)$$

where the fuzzy AND operator $\wedge$ is defined by

$$(x \wedge y)_i = min(x_i, y_i), \tag{2}$$

and the norm $|\cdot|$ is defined by

$$|\mathbf{x}| \equiv \sum_{i=1}^{m} |x_i|.$$

If γ in Eq. (1) is equal to 1, it is considered to be fast learning; otherwise, γ is set in [0,1] so that it can update the winning weight vector with a slower rate of forgetting. It is possible that the top-down searching process will fail to find a corresponding category in layer l. Under these circumstances, category W_J^l will be created and initialized by

$$W_J^l = I^l.$$

The between-layer learning subprocess can be referred to as a compression-based data-driven process [28]. That is, except for the first layer which receives an original item of input data, every other layer receives prototypes generated from its previous layer. For example, the input to BL_l with $l \geq 2$ is the output of BL_{l-1}. Let I^1 represent the input data for the first layer; then, we have

$$I^l = I^{l-1} \wedge W^{l-1}, \quad l = 2, \cdots, L, \tag{3}$$

where L is the number of layers in a CFART network. In the bottom-up learning process, one important consideration is whether or not complement coding is applied. Complement coding is a process used to solve the proliferation problem [11] (i.e., to avoid generating too many small categories). Given an n-dimensional input data $X = [x_1, \cdots, x_n]$, where x_i is in [0, 1], complement coding will extend the input data X to a $2n$-dimensional input I by

$$I = [x_1, \cdots, x_n, x_1^c, \cdots, x_n^c], \quad \text{where } x_i^c = 1 - x_i. \tag{4}$$

In a CFART network, the learning process with or without complement coding results in two different interpretations in terms of an adaptive threshold. Basically, this adaptive threshold plays an important role because it controls the number of categories generated in every layer. Therefore, the way in which its value is decided will definitely affect the design of the top-down searching process. In what follows, we shall discuss two different cases regarding the threshold value, that is, learning without or with complement coding, respectively.

2.1.1 Learning without complement coding

If complement coding is not applied, a category is selected as a candidate from the category database if its weights are very close to an item of input data and successfully pass a vigilance test, i.e.,

$$|I^l \wedge W_J^l| \geq \theta_l. \tag{5}$$

The threshold, θ_l, is used to determine an acceptable compression ratio of a category in response to an input pattern. By nature, the generated category is a maximal subset of an item of input data. The threshold value at layer l can be determined by

$$\theta_l = |I^l|\rho_l, \tag{6}$$

where ρ_l is a predetermined vigilance parameter in layer l and $0 \leq \rho_l \leq 1$. We will now explain why a CFART network is able to represent "things" in a hierarchical way. Basically, the within-layer learning subprocess is similar to that of a fuzzy ART network. Let all the vigilance parameters in different layers be the same. From Eqs. (3) and (6), the value of θ_l is proportional to the norm of the prototype developed in the previous layer. Also, this value is equal to or less than that of its previous layer since a fuzzy AND operation may reduce the norm value of an item of input data. If a smaller threshold value is chosen, fewer categories will be filtered out (Eq. (5)). Since the weights of categories are updated according to Eq. (1), the categories in higher layers are more compressed than are those in lower layers in response to an item of input data. That is, the categories in higher layers are represented in a more abstract or coarse manner.

2.1.2 Learning with complement coding

When the learning process is incorporated with complement coding, the process extends the input vector from n dimension to $2n$ dimension (Eq. (4)). If an item of input data is in a binary format, complement coding uses a combination of on/off cells to represent an input pattern [11]. The process preserves individual feature amplitudes while normalizing the total vector of on/off cells. In other words, complement coding is used to perform normalization on an item of input data. Applying this scheme, a category is generated or selected as the maximal subset of an item of input data. Therefore, for binary data, the learning process with complement coding is similar to that without complement coding except that the former can avoid the proliferation problem.

If an item of input data is in an analog format, combined with the fuzzy MIN operation, complement coding leads the learning process to generate hyperbox-shaped categories whose corners are iteratively defined through the operation of serial ($\wedge$) operators [11]. In this way, a weight vector can be interpreted as a fuzzy hyperbox. Let the weight vector w_j be written in the complement coding format, i.e.,

$$w_j = (u_j, v_j^c), \tag{7}$$

where u_j and v_j are two n-dimensional vectors in which every component is a real number ranging from 0 to 1. The weight vector represents a hyper rectangle, R_j, where u_j and v_j define two hyper corners. The size of R_j is defined as [11]

$$|R_j| \equiv |v_j - u_j|, \tag{8}$$

which is equal to the height plus the width of R_j. In order to see how a hyper rectangle looks like, Fig. 2 illustrates a two-dimensional example. In general, a threshold is used to determine the size of a hyperbox. Therefore, whenever a category is selected as a candidate from the category database, this means its weights can maximize a hyperbox that represents the input data. The selected category is, therefore, considered as a superset of an item of input data.

In the bottom-up learning process, if complement coding is applied, data input to the first layer can be interpreted as a feature point. However, by Eq. (3), the inputs to those layers with $l \geq 2$ become a hyperbox-shaped set. Fig. 3 shows four different combinations when the input, I^l, and the selected weight vector, W^l, are involved in a bottom-up between-layer learning subprocess. Three cases are considered as follows:

Case 1. If I^l is a subset of W^l(Fig. 3(a)), the norm $|I^l \wedge W^l|$ is equal to $|W^l|$.

Case 2. If W^l is a subset of I^l(Fig. 3(b)), the norm $|I^l \wedge W^l|$ is equal to $|I^l|$.

Case 3. If I^l and W^l overlap or are separated (Figs. 3(c) and 3(d)), the norm $|I^l \wedge W^l|$ is equal to or less than $|I^l| + |W^l|$.

From Eqs. (7) and (8), it is known that the size of a hyperbox increases whenever the norm value of its corresponding weights decreases. Let $W^{(new)}$ be equal to $I^l \wedge W^{(old)}$. In Cases 2 and 3, the size of a hyperbox represented by $W^{(new)}$ increases if compared with that of the old hyperbox. The size of the norm $|W^{(new)}|$ is, thus, smaller than that of $|W^{(old)}|$. Basically, this solution is the same as the compression effect in a learning process which does not apply complement coding (i.e., the norm value of weights decreases after the updating process).

In Case 1, the norm $|I^l \wedge W^l|$ is equal to $|W^l|$. In the original design of a fuzzy ART network, if the input norm $|I|$ is too large, pattern matching may fail due to

$$|I \wedge W| = |W| < |I|\rho. \tag{9}$$

Here, category W will be filtered out since the matching ratio $\frac{|W|}{|I|}$ is not large enough to pass the vigilance test. If complement coding is applied, the vigilance test in Eq. (9) still holds. The only difference is that the norm of an item of input data becomes a constant n (i.e., the dimension of an input vector). In the bottom-up learning process of a CFART, if complement coding is applied, only the item of input data for the first layer is normalized to n. Therefore, this vigilance test will hold for the learning process in the first layer. However, while propagating up to higher layers, each item of input data becomes as a hyperbox-shaped set instead of a feature point and may have a different norm value. Let I_1 and I_2 be two different inputs, where

$$|I_1^l| < |I_2^l|,$$

and

$$|I_1^l \wedge W^l| = |I_2^l \wedge W^l| = |W^l|.$$

In general, for any input I, if $|I \wedge W|$ equals $|W|$, this means $I \subset W$ (Fig. 3(a)) and that W is considered as a perfectly matched category for I. Under these circumstances, no matter what $|I_1^l|$ and $|I_2^l|$ are, W^l is always considered as the perfectly matched category of both I_1^l and I_2^l. However, if the vigilance test in Eq. (5) is adopted, it may generate a different result. That is, W^l is the matched category of I_1^l instead of I_2^l if the value of $|I_2^l|$ is much larger than that of $|I_1^l|$. To avoid this inconsistency and to consider the difference between learning with and without complement coding, the thresholding rule should be updated into the following form:

$$\theta_l = \begin{cases} 0 & \text{if } |I^l \wedge W^l| = |W^l|; \\ |I^l|\rho_l & \text{otherwise.} \end{cases} \tag{10}$$

In addition, the vigilance parameter (ρ_l) can be designed into a format such that value of θ_l will decay faster from the bottom layer to the top layer. Under these circumstances, it is guaranteed that higher layers will keep more general categories than will in lower layers.

2.2 Top-down searching

With an understanding of how the bottom-up learning process operates, we will now explain how the top-down searching scheme is design . By performing the bottom-up learning process, categories generated in higher layers will form supersets of categories generated in lower layers. That is, a hyperbox interpreted in higher layers can cover several hyperboxes located in lower layers. Given an item of input data, the searching process will locate a hyperbox in the top layer. The searching space will then be reduced to the "area" bounded by the located hyperbox. As mentioned above, a hyperbox located in layer l ($l \geq 2$) usually covers several smaller hyperboxes located in layer $l-1$. Therefore, the number of categories at a higher layer is definitely less than that of a lower layer. This kind of hierarchy provides an advantage; i.e., the searching process starts at the top layer and, therefore, can save a lot of searching time. Fig. 4 shows a searching example of a three-layered CFART network. In the example, complement coding is applied to the learning process. Five categories are generated in layer 2, i.e., $C_1^2, \cdots, C_5^2$, and two categories are generated in layer 3, i.e., C_1^3 and C_2^3. Since the input is two dimensional data, the weights of categories can be represented as boxes (Fig. 4). A dotted line rectangle represents the fuzzy boundary of a category. When a new input, I_1, is presented, the top-down searching process checks the possible fuzzy boundaries of C_1^3 and C_2^3 to see which one can include I_1. Since C_1^3 and C_2^3 together cover $C_1^2, \cdots, C_5^2$, one has to recursively check all the categories one layer below, i.e., the layer that $C_1^2, \cdots, C_5^2$ belong to. In this case, the top-down searching process will locate category C_1^3 in layer 3 and category C_2^2 in layer 2. For another input, I_2, since no category in higher layers can represent it, the searching process will stop immediately. Instead, the bottom-up learning process will be triggered to "learn" this new input pattern by creating a new category. One thing to notice is that, in the top-down searching process, every layer in a CFART network receives the same input, i.e., I (see Fig. 1). However, in the bottom-up learning process, the propagation of prototypes occurs in a sequential manner, i.e., from bottom to top. In other words, the top-down searching process can be executed in parallel, but the bottom-up learning process must be performed sequentially. From another viewpoint, we can also say that bottom-up learning is a composition process of categories and that top-down searching is a decomposition process of categories. From the learning point of view, the purpose of bottom-up learning is to "train" a matched category, and that of top-down searching is to "find" a correct category which matches an input pattern.

In a fuzzy ART network [11], the learning process incorporates a search-hypothesis test cycle so that the winning node can be located and updated. If the hypothesis test fails, a reset will ensue to start another searching cycle. It is definitely true that a series of mismatch resets in response to a single input will increase the computational load and lower the efficiency of the network. To solve this problem, an optimization approach [31, 32] is adopted. The idea is that by giving a constraint, the impossible candidates will be eliminated first, and the best matching category can be found by maximizing a measure of similarity. This can be achieved by incorporating a Hamming net structure [29, 32, 33]. Thus, the searching scheme is converted from a sequential to a parallel one. The top-down searching process of the proposed CFART network extends this single-layered parallel search to a multiple-layered parallel search and is, therefore, capable of locating a corresponding category in every layer. In order to guide the bottom-up learning, the proposed searching scheme also incorporates the characteristics of the threshold discussed in the previous subsection into a fuzzy boundary test. Fig. 5 shows the configuration of the top-down searching process in layer l. By performing the top-down searching process, the winning category index, J^l, is generated by

$$J^l = \arg_j \max_{j=1,\cdots,N_k^l} \frac{|I \wedge W_j^l|}{\alpha + |W_j^l|} \tag{11}$$

$$\text{subject to} \quad |I \wedge W_j^l| \geq \theta_l, \tag{12}$$

where N_k^l is the number of categories in the searching subset at layer l, θ_l is determined by

$$\theta_l = \begin{cases} 0 & \text{if } |I \wedge W_j^l| = |W_j^l|, \\ \min_{j=1,\cdots,N_k^{l-1}} \ |W_j^{l-1}| \ \rho_l & \text{otherwise if } l \geq 2, \\ |I^1| \ \rho_l & \text{otherwise if } l = 1, \end{cases} \tag{13}$$

and N_k^{l-1} is the number of child categories linked to category W_j^l.

The output of the MINNET in TS_l (Fig. 5) is the estimated fuzzy boundary of a category. In order to reduce the computational load needed for recursion, the output of the MINNET in TS_l can be pre-obtained through the bottom-up learning process in BL_{l-1}. A new component used to record the estimated fuzzy boundary, FB_l, is then added to represent a category in addition to the weight vector. The detailed procedure of the searching process in layer l is described as follows:

1. Present the input I.

2. If CD_l is empty, stop and trigger the bottom-up learning process.

3. For the input I and category W_j^l, compute the matching score by

$$s_j^l = \sum_{i=1}^{m} min(I_i, w_{ji}^l), \qquad 0 < j \leq N_k^l.$$

   A linear function f_θ^l is used to filter out impossible categories if their matching scores cannot pass the fuzzy boundary test. That is,

$$f_\theta^l(s_j) = \begin{cases} 0 & \text{if } s_j^l < \theta_l, \\ s_j^l & \text{if } s_j^l \geq \theta_l, \end{cases}$$

   where the threshold value θ_l is calculated by Eq. (13). If all W_j^l are filtered out, stop and trigger the bottom-up learning process.

4. Choose the category according to the selection function

$$u_j^l = \frac{f_\theta^l(s_j)}{\alpha + |W^l|},$$

where $0 < \alpha \ll 1$. The winning category index, J^l, is thus obtained by

$$J^l = \arg_j \max_j u_j^l, \quad j = 1, \cdots, N_k^l.$$

If more than one u_j^l is maximal, the output node with the smallest index is chosen to break the tie.

2.3 Construction of the searching tree

The above mentioned top-down searching process and bottom-up learning process need to be performed in association with a global searching tree. The searching tree will adapt in response to an item of input data through close interaction with the coupled processes. In general, construction of the tree is accomplished by connecting those hierarchically related categories in a bottom-up direction. Fig. 6 illustrates an example showing the construction process of a 3-layered searching tree. Suppose a searching tree has been built as shown in Fig. 6(a). In what follows, two possible situations will be discussed.

1. The structure of the searching tree remains the same(Fig. 6(b)): The top-down searching process successfully locates a matched category in every layer. The bottom-up learning process is then executed to update the weights of those matched categories in the located subtree.

2. The searching tree grows in response to an item of input data:

   (a) The category database in every layer is empty at the beginning of the learning stage. When the first pattern is input, a category needs to be created and initialized in every layer. Only a vertical link needs to be added to connect categories between layers.

(b) If a category is found in every layer except in the bottom layer, a new category in the bottom layer is created, initialized and linked to an appropriate category one layer above. Only a sibling link is needed to form a new subtree(Fig. 6(c)).

(c) In most cases, a vertical link and a sibling link are needed to form a new subtree (Fig. 6(d)). If the top-down searching process stops at layer l, then the bottom-up learning process will be triggered to create and initialize categories from the bottom layer to layer l. A sibling link is needed to connect the newly generated category in layer l and the corresponding sibling categories. Also, a vertical link is needed to connect those newly generated categories from layer 1 to layer l.

Basically, the output of a CFART network is a selected category represented in a hierarchical form. This hierarchical representation scheme represents a category with a linked path, from fine to coarse (bottom up). Given an input pattern, in a learning module, MU_l, the top-down searching process (TS_l) will try to select a best matched category located in CD_l (Fig. 1). Since the counterparts of the input image in every layer will be linked to form a subtree, the category (or class) hierarchy can be discovered. To illustrate this in an example, a simple data set with five patterns (10111 00101 10010 00010 01000) was fed into a 2-layered CFART network. The network was trained until every pattern could directly access its corresponding categories in every layer. As shown in Fig. 7, five categories were generated in MU_1 to represent five input patterns. In layer MU_2, only three categories were generated. Categories in these two layers, thus, form a hierarchical structure through linking of appropriate categories in different layers of the searching tree. Category C_1^2 in layer MU_2 represents categories C_1^1 and C_5^1 in layer MU_1 due to the similarity of the pattern (_ 0 1 _ 1). Category C_2^2 in layer MU_2 represents categories C_2^1 and C_4^1 in layer MU_1 due to the similarity of the pattern (_ 0 0 1 0). In addition to exhibiting the hierarchical relationships of categories, the prototypes of these categories can also be used to locate features that are not inherited from more general classes located at a higher level. For example, the pattern (_ 1 _ _ _) represented by Category C_3^1 in MU_1 cannot be represented by Category C_1^2 or C_2^2 in MU_2.

2.4 The learning algorithm

A complete learning algorithm of a CFART network which integrates top-down searching and bottom-up learning can be described as follows:

1. **Initialization:** Determine the number of layers, L, and the vigilance parameter, ρ_l, for every layer, l.

2. **Input:** Present a binary or analog pattern, $X = [x_1, ..., x_n]$, where x_i is in [0, 1], and n is the number of input nodes (i.e., the dimension of the input vector).

3. **Input coding:** If complement coding is applied, the input to the network is extended from dimension n to m by

$$I = [x_1, \cdots, x_n, x_1^c, \cdots, x_n^c], \quad \text{where } x_i^c = 1 - x_i \text{ and } m = 2n;$$

   otherwise, $I = X$ and $m = n$.

   Let I^1 be equal to I.

4. **Top-down searching:**

   For every MU_l, $(l = L, l \geq 1, l = l - 1)$ Do

   4.1 If CD_l is empty, set SL to be l and J^l to be NULL. Break the top-down searching process and Goto (5).

   4.2 Calculate the matching scores by

$$s_j^l = \sum_{i=1}^{m} min(I_i, w_{ji}^l), \quad 0 < j \leq N_k^l,$$

   where N_k^l is the number of candidate sibling categories.

4.3 Perform a fuzzy boundary test to filter out impossible categories by

$$f_\theta^l(s_j) = \begin{cases} 0 & \text{if } s_j^l < \theta_l, \\ s_j^l & \text{if } s_j^l \geq \theta_l, \end{cases}$$

where

$$\theta_l = \begin{cases} 0 & \text{if } |I \wedge W_j^l| = |W_j^l|, \\ FB_l\ \rho_l & \text{otherwise if } l \geq 2, \\ |I|\ \rho_l & \text{otherwise if } l = 1, \end{cases}$$

and FB_l will be calculated in the bottom-up learning process.

4.4 If all categories are filtered out, set SL to be l and J^l to be NULL. Break the top-down searching process and Goto (5).

4.5 Choose a category according to the selection function

$$u_j^l = \frac{f_\theta^l(s_j)}{\alpha + |W^l|},$$

where $0 < \alpha \ll 1$. The winning category index, J^l, is thus obtained by

$$J^l = \arg_j \max_j\ u_j^l, \quad j = 1, \cdots, N_k^l.$$

If more than one u_j^l is maximal, the output node with the smallest index is chosen to break the tie.

5. **Bottom-up learning:**

For every MU_l, $(l = 1, l \leq L, l = l + 1)$ Do

5.1 Between-layer learning:

The input for higher layers MU_l $(l \geq 2)$ is calculated by

$$I^l = I^{l-1} \wedge W_J^{l-1},$$

where $l \geq 2$ and W_J^{l-1} is the weight of a located category in MU_{l-1}.

5.2 Within-layer learning:

5.2.1 If J^l is NULL, set J^l to be $N_k^l + 1$. A new category is generated and initialized by

$$W_J^l = I^l.$$

5.2.2 If the J^lth category in CD_l is chosen, W_J^l is updated by

$$W_J^{(new)} = W_J^{(old)} + \gamma(I^l \wedge W_J^{(old)} - W_J^{(old)}),$$

where γ is set in [0,1]. If γ is equal to 1, the learning is considered to be fast learning; otherwise, the winning weight vector is updated with a slower rate of forgetting.

5.3 If $1 \leq l < L$, the fuzzy boundary of the located category in one layer above is determined by

$$FB_{l+1} = \min_{j=1,\cdots,N_k^l} |W_j^l| \; \rho_{l+1}.$$

6. **Construction of searching tree:**

6.1 For every MU_l, $(l = 1, l \leq SL, l = l + 1)$ Do

Build a vertical link to connect all W_J^l.

6.2 Build a sibling link at layer SL to connect W_J^{SL} with its corresponding sibling categories.

7. **Goto** (2) until the network is stable, i.e., no new category in CD_l is created and the weights are not further changed.

3 Experimental Examples

Example 1: In this experiment, we used 340 two-dimensional feature points as a training data set to test the effectiveness of the proposed CFART network. These data were spread into seven Gaussian distributions, and among these distributions four were overlapped to form two bigger clusters. The test data were related to object recognition by considering that features of objects were extracted by a preprocessing process and then transformed into an appropriate feature vector form. In this example, each two-dimensional data point represented a feature vector with two feature components. Fig. 8(a) shows the scatter plot of the data used in this example. The number of layers required for the CFART network used in the experiment was set to be three. The vigilance parameters were set to be $\rho_1 = 0.8$, $\rho_2 = 0.60$ and $\rho_3 = 0.35$. Further, we applied complement coding in the learning process to normalize the input data. After the learning process of the network became stable (i.e., every input could access a matching category in each layer, and the weights of a selected category were not changed), the cascaded relationships of different categories represented by multi-layers of a CD network were as shown in Fig. 8(b). Figs. 8(c), (d) and (e) show a series of generated categories at different layers of the CD. It is obvious that categories in higher layers formed a superset of categories in lower layers (Fig. 8(f)). Therefore, we can prove that categories in higher layers express a more general view than do those in lower layers.

Example 2: The superquadrics modeling scheme is one of the most powerful 3D object representation schemes in use nowadays. Superquadrics can intuitively be thought of as lumps of clay that can be deformed and glued together into object models [34]. Mathematically, superquadrics form a parameterized family of shapes. The most commonly used superquadric surface is a superellipsoid, which is determined by

$$X(\eta, \omega) = \begin{bmatrix} a_1 cos^{\varepsilon_1}(\eta) cos^{\varepsilon_2}(\omega) \\ a_2 cos^{\varepsilon_1}(\eta) sin^{\varepsilon_2}(\omega) \\ a_3 sin^{\varepsilon_1}(\eta) \end{bmatrix},$$

where

$$\begin{aligned} -\pi/2 &\leq \eta &\leq \pi/2, \\ -\pi &\leq \omega &< \pi. \end{aligned}$$

The angle parameters η and ω correspond to the latitude and longitude angles of a vector X expressed in a spherical coordinate. The parameters a_1, a_2, and a_3 correspond to the size of a superquadric object in the x, y, and z directions, respectively. The parameters ε_1 and ε_2 are the shape parameters in the latitudinal and longitudinal directions, respectively. In this experiment, 16 superquadric objects generated in [35] were used to test the performance of the proposed CFART network. The generated superquadric objects and their corresponding parameters are shown in Fig. 9. A 4-layered CFART network was used. The vigilance parameters required for each layer from coarse to fine were $\rho_4 = .7$, $\rho_3 = .8$, $\rho_2 = .9$ and $\rho_1 = 1$, respectively. Since the network requires the value of an input ranging between 0 and 1, the values of a_1, a_2 and a_3 were divided by 40, respectively, and the values of ε_1 and ε_2 were divided by 2, respectively. The network was trained until every input superquadric could access a matching category directly in every layer. Fig. 10 shows the categories generated in every layer. In layer 1 (Fig. 10(d)), 16 categories were generated and each contained only one object due to the lack of error tolerance allowed, i.e., $\rho_1 = 1$. The weight of a category with complement coding represents a hyper-box formed by a minimum and a maximum hyper-corner [11]. In the application, a hyper-corner represented a superquadric object. Figs. 10(c1), (b1) and (a1) show 10, 5 and 2 superquadric objects represented by the minimum hyper-corners of categories generated in layers 2, 3 and 4, respectively. Figs. 10(c2), (b2) and (a2) show 10, 5 and 2 superquadric objects represented by the maximum hyper-corners of categories generated in layers 2, 3 and 4, respectively. Therefore, each category in every layer represented a set of superquadric objects bounded by a hyper-box. The coarse-to-fine representation with hierarchical links built in the searching tree of the network is shown in Fig. 11. Here, a category is illustrated by an object which is the fusion of the corresponding minimum and maximum hyper-corners of weights. For example, categories $C_1^2, \cdots, C_{10}^2$ in Fig. 11 were illustrated by averaging the superquadric objects in Fig. 10(c1) and (c2). The multi-resolutional representation of an object can be found in a subtree, e.g., $C_1^4 - C_2^3 - C_3^2 - C_4^1$. It is obvious that categories generated in higher layers have more general views than do those in lower layers. In layer 1, 16 categories were generated to specifically represent 16 test objects. Upon going up to layer 2, 10 categories with more general shapes were generated due to the coarser resolution of layer 2 compared to that of layer 1. For example, category C_5^2 subsumed categories C_6^1 and C_{10}^1 due to their similar shapes. Continuing to go up, because of the coarser resolution, the number of categories was reduced to 5 and 2 in layers 3 and 4, respectively. From the above hierarchical representation scheme, it is obvious that a category can be efficiently located by top-down searching through the hierarchical links in the searching tree.

4 Discussion

As described in the previous section, the proposed CFART network provides a hierarchical representation scheme. In the representation scheme, input data are represented by sets of different sized, hyperbox-shaped weights located in different layers. The size of the hyperboxes varies from small to large (fine-to-coarse), depending on an adaptive threshold. Usually, the adaptive threshold decreases its value from lower layers to higher layers such that fine-to-coarse learning can be achieved. There are two ways to decrease the threshold value. One is to reduce the vigilance value from bottom to top. The other is to set all the vigilance values to be equal, but since the norm value of compressed input data has a tendency to decrease from lower layers to higher layers, fine-to-coarse learning can still be achieved if the number of layers is large enough. In our scheme, we incorporate both methods to speed up the learning process. In what follows, we shall discuss how the optimal number of layers of a CFART network in a general situation is decided.

If the input data is in binary format, the bottom-up learning of a CFART is functionally equivalent to a hierarchical ART (HART) [28] network. The maximum number of layers, L_{max}, has an upper bound if complement coding is applied. Let n be the number of layers, K be the dimension of the input vector, and ρ be the vigilance value for all all layers. The maximum number of layers can be estimated by [28]

$$L_{max} = \lfloor n + 1 \rfloor,$$

where

$$n > -\frac{logK}{log\rho}.$$

However, if the input data are in analog form, then the above derivation does not hold. Instead, one has to determine the number of layers according to the range of the data or the desired degree of variation of the fine-to-coarse representation. Basically, if the number of input data is finite, the size of the hyperbox-shaped weights in the top layer cannot exceed the range formed by the minimum and the maximum of the input data. Under the circumstances, the number of layers is definitely finite. One possible solution to be investigated in future work for the above mentioned problem is to use an adaptive method to approximate the number of layers. The adaptive method could be used to increase the number of layers one by one or to determine this number by pruning a given big number. The optimum number of layers of a CFART network can, thus, be judged by checking whether the network can recognize the input data and correctly represent their complex structure without losing the efficiency of the learning scheme.

5 Concluding Remarks

We have successfully built a cascade fuzzy ART neural network. The cascade networks are able to create new categories in each layer to learn the input data through interaction of coupled parallel top-down searching and a sequential bottom-up learning process. The hierarchical relationships of input data are then exposed through a path in the searching tree. The cascade fuzzy ART neural network is capable of learning complex and structured input data. Further, it can act as an extensible database in a model-based object recognition system and provide a fine-to-coarse view of input data.

References

[1] P. J. Besl and R. C. Jain, "Three-dimensional Object Recognition", *ACM Computing Surveys*, vol. 18, no. 1, pp. 75–145, Mar. 1985.

[2] R. T. Chin and C. R. Dyer, "Model-based Recognition in Robot Vision", *ACM Computing Surveys*, vol. 18, no. 1, pp. 67–108, 1986.

[3] F. Arman and J. K. Aggarwal, "Model-based Object Recognition in Dense-Range Images - A Review", *ACM Computing Surveys*, vol. 25, no. 1, pp. 5–43, Mar. 1993.

[4] W. C. Lin, H. Y. Liao, C. K. Tsao, and T. Lingutla, "A Hierarchical Multiple-View Approach to Three-Dimensional Object Recognition", *IEEE Transactions on Neural Networks*, vol. 2, no. 1, pp. 85–92, 1991.

[5] H. Y. Liao, C. C. Liang, and W. C. Lin, "ART-1 Neural Network for Reducing Search Space in 3-D Object Recognition Using Multiple Views", *International Journal of Neural Networks*, vol. 2, pp. 81–89, Jun.-Dec. 1991.

[6] G. Gindi, E. Mjolsness, and P. Anandan, "Neural Networks for Model Based Recognition", in *Neural Networks Concepts, Application, and Implementations*, P. Antognetti and V. Milutinović, Eds., vol. III, chapter 7, pp. 144–173. Prentice-Hall, Inc., 1991.

[7] G. A. Carpenter and S. Grossberg, "A Massively Parallel Architecture for a Self-Organizing Neural Pattern Recognition Machine", *Computer Vision, Graphics, and Image Processing*, vol. 37, pp. 54–115, 1987.

[8] G. A. Carpenter and S. Grossberg, "ART2: Self-Organizing of Stable Category Recognition Codes for Analog Input Patterns", *Applied Optics*, vol. 26, no. 23, pp. 4919–4930, Dec. 1987.

[9] G. A. Carpenter, S. Grossberg, and D. B. Rosen, "ART 2-A: An Adaptive Resonance Algorithm for Rapid Category Learning and Recognition", *Neural Networks*, vol. 4, pp. 493–504, 1991.

[10] G. A. Carpenter and S. Grossberg, "ART 3: Hierarchical Search Using Chemical Transmitters in Self-Organizing Pattern Recognition Architectures", *Neural Networks*, vol. 3, pp. 129–152, 1990.

[11] G. A. Carpenter, S. Grossberg, and D. B. Rosen, "Fuzzy ART: Fast Stable Learning and Categorization of Analog Patterns by an Adaptive Resonance System", *Neural Networks*, vol. 4, pp. 759–771, 1991.

[12] T. Kohonen, *Self-Organization and Associative Memory*, Springer-Verlag, Berlin, 3 edition, 1989.

[13] G. A. Carpenter and W. D. Ross, "ART-EMAP: A Neural Network Architecture for Object Recognition by Evidence Accumulation", *IEEE Transactions on Neural Networks*, vol. 6, no. 4, pp. 805–818, Jul. 1995.

[14] D. Bullock, G. A. Carpenter, and S. Grossberg, "Self-Organizing Neural Network Architectures for Adaptive Pattern Recognition and Robotics", in *Neural Networks Concepts, Apllications, and Implementations*, P. Antognetti and V. Milutinović, Eds., vol. I, chapter 3, pp. 33–53. Prentice-Hall, Inc., 1991.

[15] S. Grossberg, Ed., *Neural Networks and Artificial Intelligence*, The MIT Press, 1988.

[16] J. C. Rajapakse, O. G. Jakubowicz, and R. S. Acharya, "A Real Time ART-1 Based Vision System for Distortion Invariant Recognition and Autoassociation", in *Proceedings of International Joint Conference on Neural Networks*, Washington, D. C., Jan. 1990, pp. 298–301.

[17] P. L. Galindo and T. Michaux, "An Improved Competitive Learning Algorithm Applied to High Level Speech Processing", in *Proceedings of International Joint Conference on Neural Networks*, Washington, D. C., Jan. 1990, pp. 142–146.

[18] K. Fukushima and S. Miyake, "Neocognitron: A New Algorithm for Pattern Recognition Tolerant of Deformations and Shifts in Position", *Pattern Recognition*, vol. 15, no. 6, pp. 455–469, 1982.

[19] K. Fukushima, S. Miyake, and T. Ito, "Neocognitron: A Neural Network Model for a Mechanism of Visual Pattern Recognition", *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-13, no. 5, pp. 826–834, 1983.

[20] L. Breiman, J.H. Friedman, R.A. Olshen, and C.J. Stone, *Classification and Regression Trees*, Wadsworth, Belmont, California, 1984.

[21] K. Fukushima, S. Miyake, and T. Ito, "Neocognitron: A Hierarchical Neural Network Capable of Visual Pattern Recognition", *Neural Networks*, vol. 1, pp. 119–130, 1988.

[22] S. R. Safavian and D. Landgrebe, “A Survey of Decision Tree Classifier Methodology”, *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 21, no. 3, pp. 660–674, May/Jun. 1991.

[23] M. Golea and M. Marchand, “A Growth Algorithm for Neural Network Decision Trees”, *Europhysics Letters*, vol. 12, no. 3, pp. 205–210, Jun. 1990.

[24] H. L. Hung and W. C. Lin, “Dynamic Hierarchical Self-organizing Neural Networks”, in *Proceedings of IEEE International Conference on Neural Networks*, Orlando, FL., Jun. 28 - Jul. 2 1994, vol. II, pp. 627–632.

[25] D. I. Choi and S. H. Park, “Self-Creating and Organizing Neural Networks”, *IEEE Transactions on Neural Networks*, vol. 5, no. 4, pp. 561–575, Jul. 1994.

[26] T. Li, Y. Y. Tang, and L. Y. Fang, “A Structure-parameter-adaptive (SPA) Neural Tree for the Recognition of Large Character Set”, *Pattern Recognition*, vol. 28, no. 3, pp. 315–329, 1995.

[27] B. A. Draper, C. E. Brodley, and P. E. Utgoff, “Goal-Directed Classification Using Linear Machine Decision Trees”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 16, no. 9, pp. 888–893, 1994.

[28] G. Bartfai, “An ART-based Modular Architecture for Learning Hierarchical Clusterings”, Tech. Rep. CS-TR-95/3, Victoria University of Wellington, Feb. 1995, To appear on *Neurocomputing.*

[29] H. L. Hung, S. J. Lin, H. Y. M. Liao, and W. C. Lin, “Cascade Fuzzy Adaptive Hamming Net: An Extensible Database for Object Recognition Systems”, in *Proceedings of International Symposium on Artificial Neural Networks*, Taiwan, Dec. 1995, pp. C2.13–18.

[30] H. L. Hung, H. Y. M. Liao, S. J. Lin, W. C. Lin, and K.-C. Fan, “Cascade Fuzzy ART: A New Extensible Database for Model-Based Object Recognition”, in *Proceedings of SPIE on Visual Communications and Image Processing*, Orlando, Florida, Mar. 1996, To appear.

[31] M. J. Healy, T. P. Caudell, and S. D. G. Smith, “A Neural Architecture for Pattern Sequence Verification Through Inferencing”, *IEEE Transactions on Neural Networks*, vol. 4, no. 1, pp. 9–20, Jan. 1993.

[32] C. A. Hung and S. F. Lin, “Adaptive Hamming Net: A Fast-Learning ART 1 Model Without Searching”, *Neural Networks*, vol. 8, no. 4, pp. 605–618, 1995.

[33] Y. H. Pao, *Adaptive Pattern Recognition and Neural Networks*, Addison-Wesley Publishing Company, Inc., 1989.

[34] R. M. Haralick and L. G. Shapiro, *Computer and Robot Vision*, vol. II, Addison-Wesley Publishing, Inc., U.S.A., 1993.

[35] Y. T. Liu, H. Y. Liao, L. H. Chen, and F. C. Lin, “A New Metric Between Superquadric Models for 3-D Object Recognition”, in *Proceeding of IPPR Conference on Computer Vision, Graphics and Image Processing*, Taiwan, Aug. 1993, pp. 333–340.

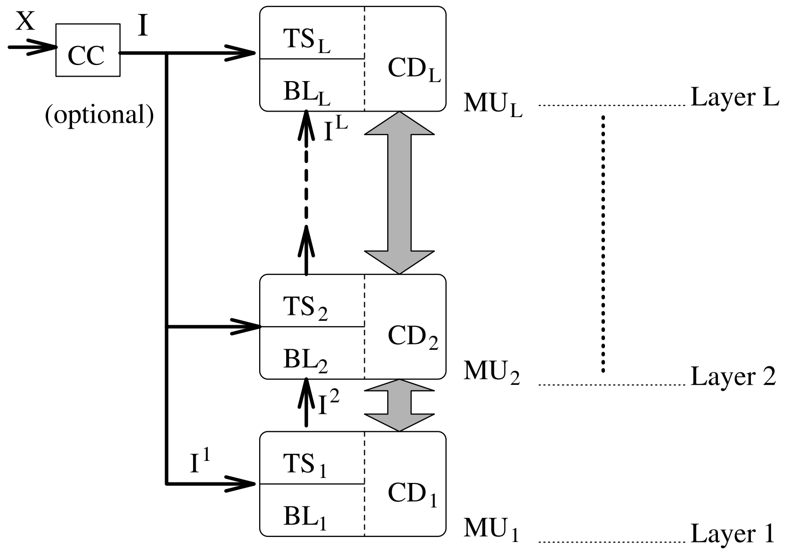


Figure 1: The architecture and data flow of a cascade fuzzy ART (CFART) network. (CC: Complement Coding, TS: Top-Down Searching, BL: Bottom-Up Learning, CD: Category Database, MU: Module Unit)

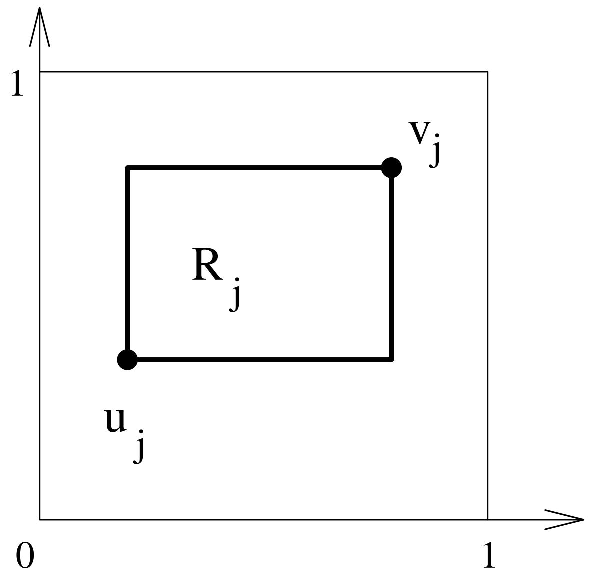
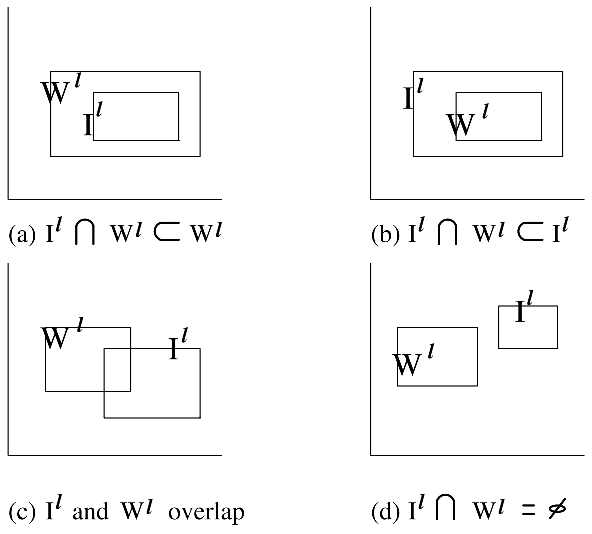


Figure 2: A two dimensional hyper rectangle: the weight vector w_j (u_j, v_j^c) in complement coding form has a geometric interpretation as a rectangle R_j with corners $(u_j,\, v_j)$.



W^l
I^l
(a) $I^l \cap W^l \subset W^l$
I^l
W^l
(b) $I^l \cap W^l \subset I^l$
W^l
I^l
(c) I^l and W^l overlap
I^l
W^l
(d) $I^l \cap W^l = \not\subset$

Figure 3: Four combinations of $I^l \wedge W^l$.

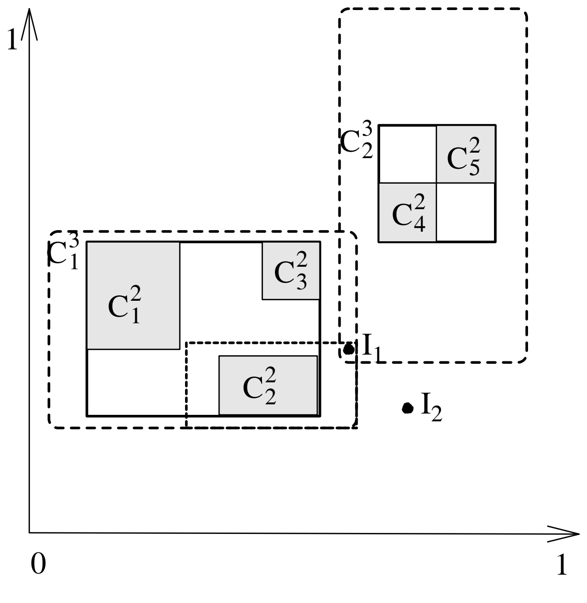


Figure 4: A searching example of a three-layered CFART network: 5 categories $(C_1^2, \cdots, C_5^2)$ are in layer 2, and 2 categories (C_1^3 and C_2^3) are in layer 3. I_1 and I_2 are two input data. Dotted line rectangles represent the fuzzy boundaries of categories.

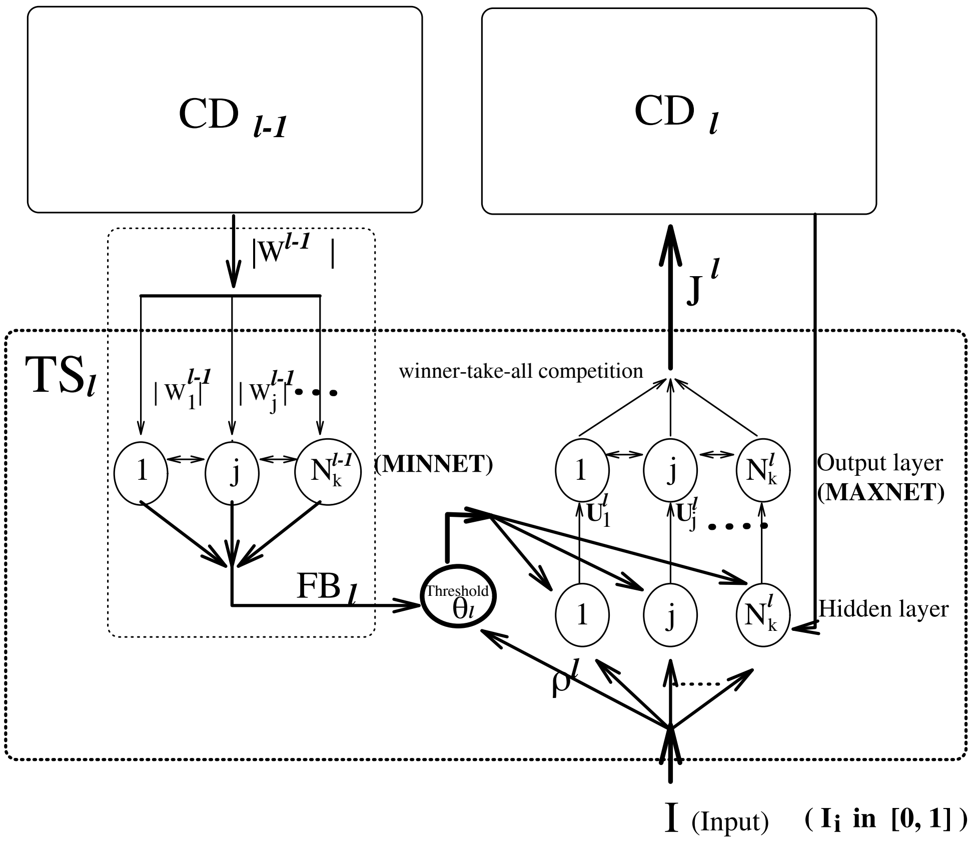


Figure 5: The configuration of the top-down searching process in layer l.

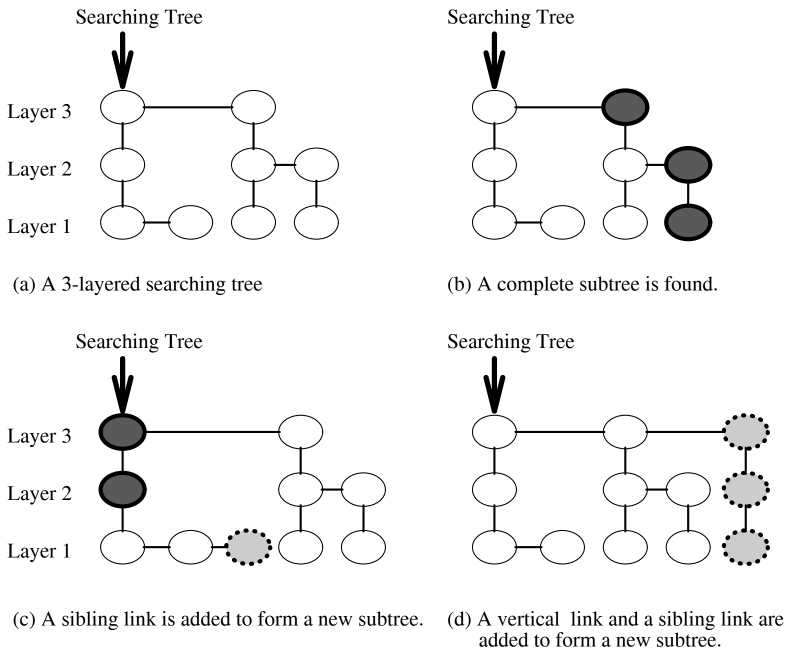

Searching Tree
Searching Tree
Layer 3
Layer 2
Layer 1
(a) A 3-layered searching tree
(b) A complete subtree is found.
Searching Tree
Searching Tree
Layer 3
Layer 2
Layer 1
(c) A sibling link is added to form a new subtree.
(d) A vertical link and a sibling link are added to form a new subtree.

Figure 6: The construction process of a three-layered searching tree: a solid shadowed circle means a category located by the top-down searching process, and a dotted shadowed circle means a category created by the bottom-up learning process.

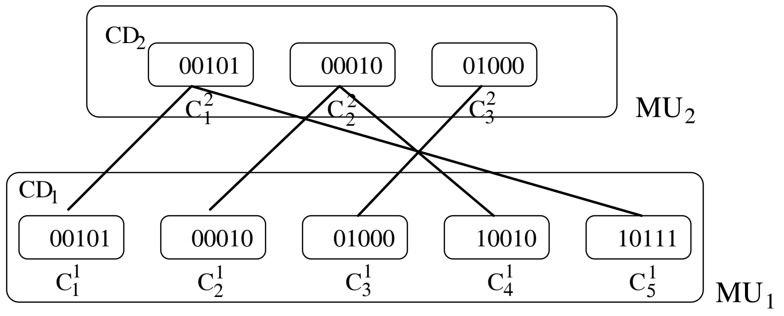
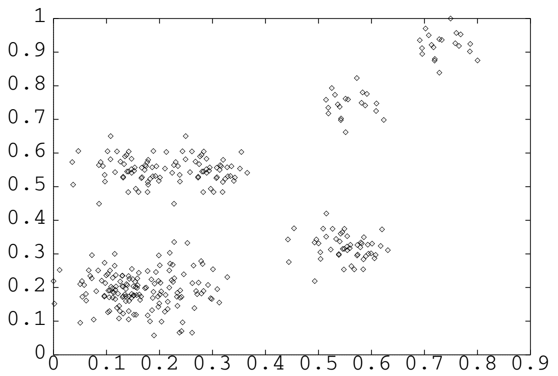
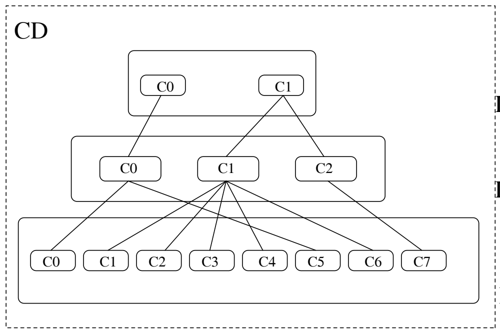


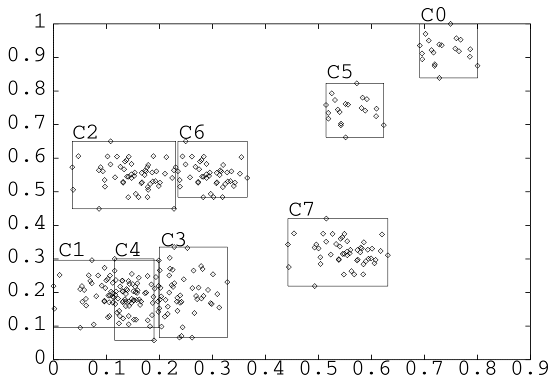
Figure 7: Hierarchical categories in a 2-layered CFART network with five input patterns.



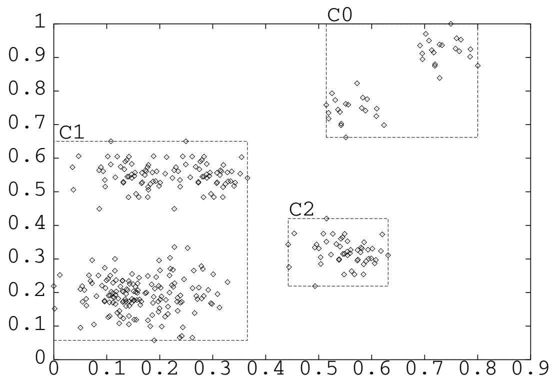
(a) 2-dimensional test data



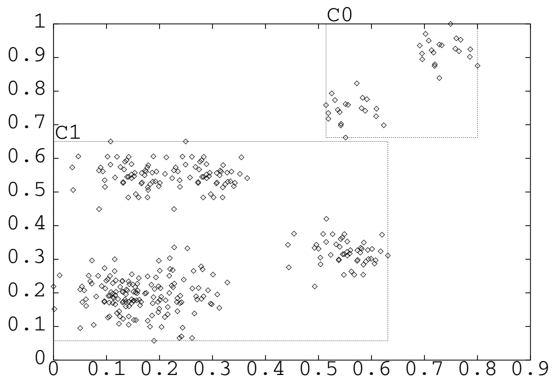
(b) Cascade relationships of variant categories represented by multi-layers of a CD network



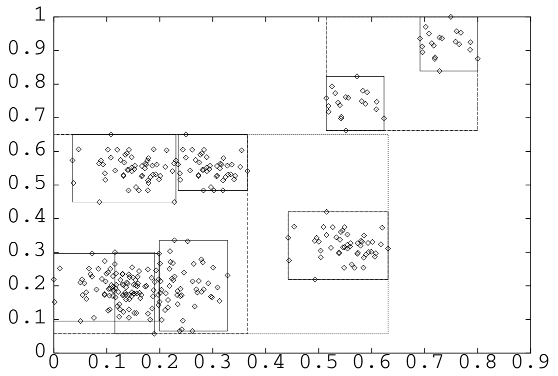
(c) Generated categories in MU_1



(d) Generated categories in MU_2

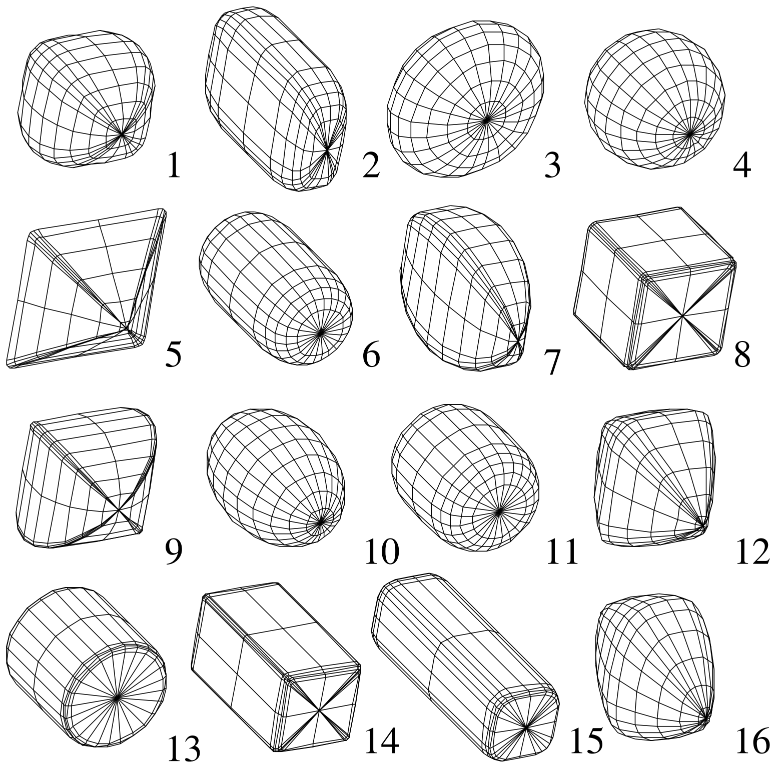


(e) Generated categories in MU_3



(f) Categories in higher layers form supersets of categories in lower layers.

Figure 8: A 2-dimensional case with analog input fed into a 3-layered CFART network. ($\rho_1 = .8$, $\rho_2 = .6$, $\rho_3 = .35$)



	a_1	a_2	a_3	ε_1	ε_2
1	20.14	20.14	22.45	1.0	0.5
2	10.11	21.24	32.37	0.5	0.5
3	27.81	27.81	13.90	1.0	1.0
4	22.07	22.07	22.07	1.0	1.0
5	25.69	25.69	25.69	2.0	2.0
6	16.92	16.92	28.63	0.5	1.0
7	11.56	22.50	33.44	1.0	0.3
8	17.86	17.86	17.86	0.1	0.1
9	20.39	20.39	20.39	1.0	0.1
10	19.28	19.28	28.92	1.0	1.0
11	19.36	19.36	21.89	0.5	1.0
12	18.47	23.33	30.78	1.6	0.4
13	19.33	19.33	19.33	0.1	1.0
14	14.34	14.34	27.66	0.1	0.1
15	12.68	12.68	38.04	0.1	0.5
16	16.48	24.73	32.97	1.5	0.7

Figure 9: The 16 test superquadric objects and their corresponding parameters.

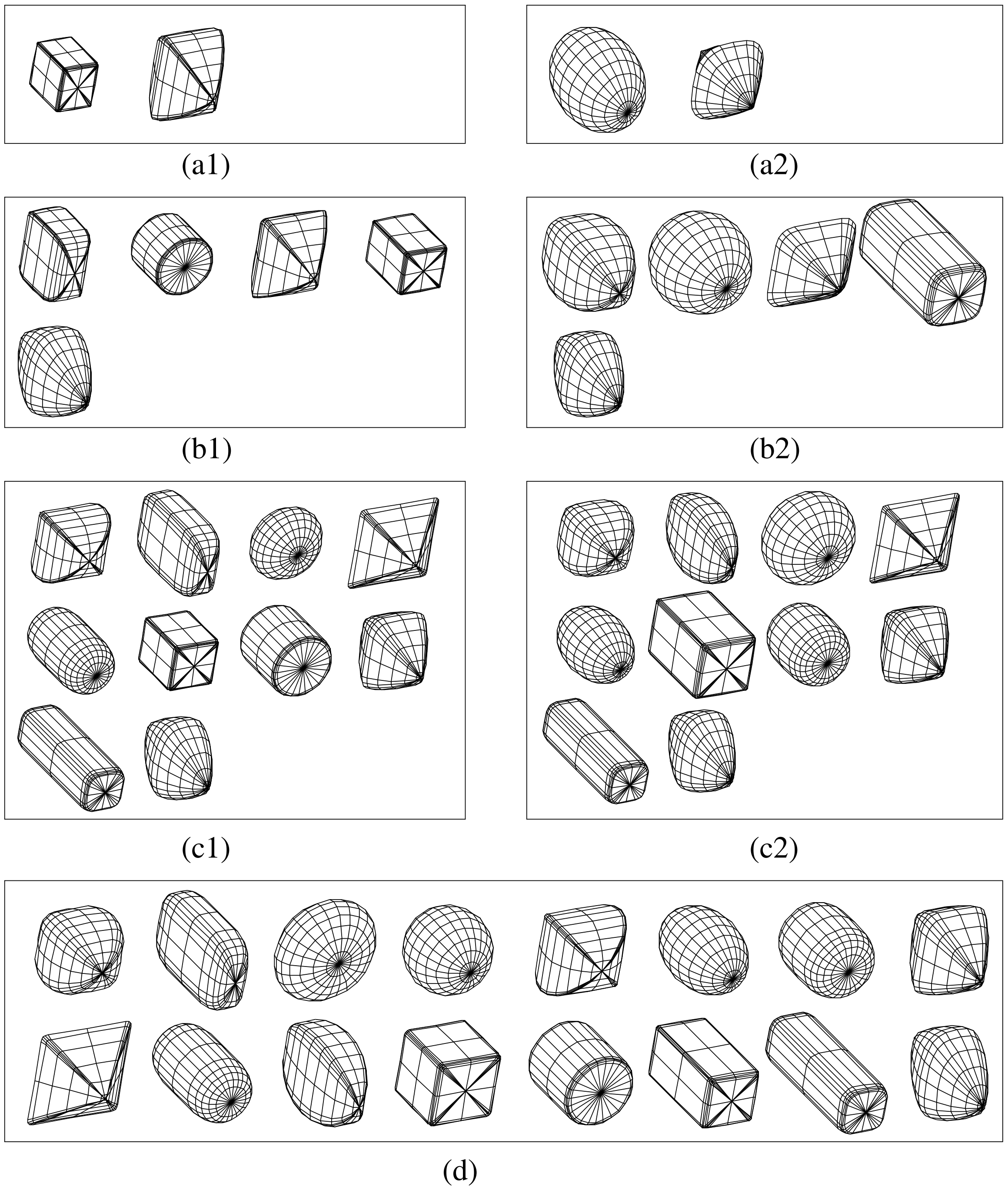


Figure 10: The generated categories in a 4-layered CFART network: (a1),(b1) and (c1) show 2, 5 and 10 superquadric objects represented by the minimum hyper-corners of categories generated in layers 4, 3 and 2, respectively. (a2),(b2) and (c2) show 2, 5 and 10 superquadric objects represented by the maximum hyper-corners of categories generated in layers 4, 3 and 2, respectively. (d) shows superquadric objects represented by categories generated in layer 1.

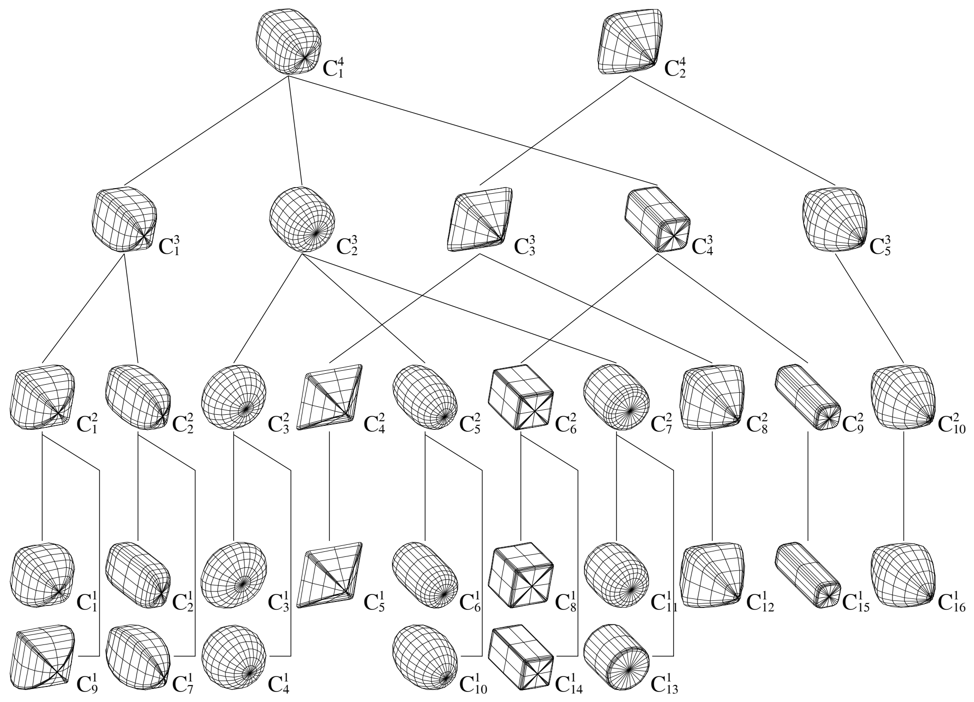


Figure 11: The coarse-to-fine representation with hierarchical links of 16 test superquadric objects.