Journal of Intelligent Manufacturing (1997) **8**, 203–214

Feature recognition using ART2: a self-organizing neural network

KISHORE LANKALAPALLI*, SUBRATA CHATTERJEE and T.C. CHANG

1287 Grissom Hall, School of Industrial Engineering, Purdue University, West Lafayette, IN 47907, USA

Received November 1995 and accepted June 1996

A self-organizing neural network, ART2, based on adaptive resonance theory (ART), is applied to the problem of feature recognition from a boundary representation (B-rep) solid model. A modified face score vector calculation scheme is adopted to represent the features by continuous-valued vectors, suitable to be input to the network. The face score is a measure of the face complexity based upon the convexity or concavity of the surrounding region. The face score vector depicts the topological relations between a face and its neighbouring faces. The ART2 network clusters similar features together. The similarity of the features within a cluster is controlled by a vigilance parameter. A new feature presented to the net is associated with one of the existing clusters, if the feature is similar to the members of the cluster. Otherwise, the net creates a new cluster. An algorithm of the ART2 network is implemented and tested with nine different features. The results obtained indicate that the network has significant potential for application to the problem of feature recognition.

Keywords: Feature recognition, feature representation, neural networks, ART

1. Feature recognition

To automate manufacturing there is a need to establish an interface between computer-aided design (CAD) and computer-aided process planning (CAPP). The designs need to be analyzed for functional considerations such as manufacturability and assemblability in order to generate effective process plans. Solid models depict the complete geometric and topological information of engineering components. The two prominent solid models known are *boundary representation* (B-rep) and *constructive solid geometry* (CSG). The B-rep model of a solid contains the geometric and topological information of a solid in terms of low-level entities such as faces, loops, edges, vertices and the relationships between these geometric entities. This low-level information cannot be directly used for analyzing designs for functional considerations. Feature recognition is the most promising approach to bridge the gap between solid models and process planners. The problem of feature recognition involves recognition of higher-level geometric entities, termed *features*, from the lower level entities in the solid model.

1.1. *Classification of features*

Features are physically distinguishable geometric elements of engineering significance. It is the faces of a solid component that constitute a feature. From this perspective, features can be classified as simple, intermediate and complex. Each face of a solid, by itself, is a *simple feature*. Planar, convex and concave surfaces are simple features. *Intermediate features* are a combination of two or more simple features. The topological relationship between the simple features determines the intermediate feature. Example of intermediate features are slots, pockets, and holes. The third category, *complex features*, results from the interaction of multiple intermediate features. A cross-slot belongs to this category. To generate a complete process plan all the simple, intermediate and complex features should be recognized from the solid model. The information about a simple feature can be directly accessed from the solid model. Researchers have succeeded, to some extent, in recognizing intermediate features using various methods. It is the complex features that make the problem of feature recognition quite difficult, and research is in progress in this area.

*Author to whom all correspondence should be addressed

0956-5515

1.2. *Review of feature recognition research*

Feature recognition from B-rep models has been studied by a number of researchers since the early 1980s. The various approaches proposed include: a syntactic pattern recognition approach by Kyprianou (1980) and Choi (1982); a rule-based approach by Kung (1984) and Henderson (1984); and a graph-based approach by Joshi and Chang (1988). A detailed discussion of these approaches and a summary of feature recognition studies until 1987 were reported by Chang (1990). A number of studies based on the above approaches have been reported since 1987, some of which are: rule-based systems by Hwang (1988) and Vanderbrande (1990); and graph-based systems by Sakurai and Gossard (1988), and Chuang (1991). A graph-based system using a differential depth filter technique to reduce the search space was proposed by Gadh and Prinz (1991). The first-known neural networks approach using a perceptron for feature recognition was proposed by Hwang and Henderson (1992). Unfortunately, none of the systems reported is perfect, and all of them have limitations for practical use. The drawbacks of these systems can be attributed to the methodology as well as to the implementation algorithms.

1.3. *Feature recognition using a perceptron*

An approach for feature recognition from a B-rep solid model using a single-layer perceptron was proposed by Hwang and Henderson (1992). The perceptron is a pattern classifier with supervised training, which can classify only linearly separable patterns. Supervised training involves presenting a sequence of training vectors, or patterns, each with an associated target vector, until the network adapts its weights according to a learning algorithm. Because the input to a perceptron or any other neural network is usually a vector of real numbers, an empirical face score vector calculation method to represent features was adopted. The network was trained to recognize intermediate features such as a pocket, a slot and a through-hole. When presented with partial features, the network was able to recognize them with certain confidence, but it failed to recognize complex features, such as a cross-slot. When the network was retrained by including the cross-slot feature in the training set, it was able to identify the feature with a high level of confidence. Supervised neural networks such as the perceptron perform well when the training set used is a good representative of the entire domain. In problems such as feature recognition it may not be possible to train the net with all possible features.

1.4. *Current approach*

Self-organizing neural networks based on adaptive resonance theory (ART) may overcome the problems associated with using a perceptron for feature recognition. The ART networks cluster similar features together without supervision. The degree of similarity of patterns placed on the same cluster can be controlled. Clustering of features is unique and always guaranteed. When the network is presented with an 'unfamiliar' feature – a feature that does not belong to any of the previously formed clusters – it creates a new cluster with the new feature as an exemplar for the cluster, unlike the perceptron, which fails to do so. In this paper, the feasibility of using a self-organizing neural network, ART2, for feature recognition is studied. The face score vector method proposed by Hwang and Henderson (1992) is modified to obtain a suitable feature representation scheme. An algorithm of ART2 network is implemented in MATLAB^{TM} and the results obtained are reported.

In the next section, a brief overview of adaptive resonance theory is given. A detailed description of the face score vector calculation for nine different features is presented in Section 3. In Section 4, a step-by-step algorithm for ART2 is provided. In Section 5, the results of the network, when presented with the nine features, are reported, followed by a discussion.

2. Adaptive resonance theory

Adaptive Resonance Theory (ART) was developed by Carpenter and Grossberg (1987a). The network ART1 was designed for clustering binary vectors, and later ART2, which clusters continuous-valued vectors, was developed (Carpenter and Grossberg, 1987b). These networks cluster inputs by unsupervised learning. Each time a pattern is presented, an appropriate cluster unit is chosen, and the cluster's weights are adjusted to let the cluster unit learn the pattern. The degree of similarity of patterns placed in the same cluster is controlled by a reset mechanism via a *vigilance parameter*. A new feature presented to the net is associated with one of the existing clusters, if the feature is similar to the members of the cluster. Otherwise, the net creates a new cluster.

In networks with supervised learning algorithms, such as a perceptron or a backpropagation network, an input training set is presented sequentially until the network finishes learning the entire training set. When an additional pattern is presented to the network, the network has to be completely retrained with all the training patterns. If the network learns an additional pattern alone, in the process, it forgets the previous learning and classifies the previously learned patterns incorrectly. The ability of a network to learn a new pattern is called *plasticity*, and the ability for the new learning not to be affected by the previous learning is called *stability*. The quest for *stable-plastic* networks led to the development of ART networks. According to Carpenter (1989):

ART networks are designed, in particular, to resolve the *stability–plasticity dilemma*: they are stable enough to preserve significant past learning, but nevertheless remain adaptable enough to incorporate new information whenever it might appear.

2.1. *Basic architecture*

The basic architecture of an adaptive resonance network involves three groups of neurons: input processing units (F1 layer), cluster units (F2 layer), and reset units (Fausett, 1994). The F1 layer consists of two parts: the input units (F1(a) layer) and the interface units (F1(b) layer). The interface units combine signals from the input units and the F2 layer to compare the similarity of the input signal to the weight vectors of the cluster units. A schematic of the basic architecture is shown in Fig. 1.

There are two sets of connections between the layers F1(b) and F2. The bottom-up weights connecting F1(b) to F2 are denoted by b_{ij} and the top-down weights connecting F2 to F1(b) are designated t_{ji}. The cluster unit with the largest net input becomes the candidate to learn the input vector. The activations (outputs) of all other F2 units are set to zero. Whether or not this cluster unit is allowed to learn the input vector depends on how similar its top-down weight vector is to the input vector. This decision is made by the reset unit, based on signals it receives from the input and interface units of the F1 layer and a vigilance parameter. If the cluster unit is not allowed to learn, it is inhibited, and a new cluster unit is selected as the candidate. In this paper, the cluster units correspond to features and the input vectors correspond to face score vectors.

2.2. *Vigilance parameter*

The criterion for an adequate match between an input pattern and a chosen cluster (feature) is determined by a vigilance parameter, which ranges between 0 and 1. All other things being equal, higher vigilance imposes a stricter matching criterion, which in turn partitions the input set into finer clusters. Lower vigilance tolerates greater mismatches, leading to coarser clusters. The choice of the vigilance parameter is critical to the performance of a ART network, but there are no guidelines for setting the value of vigilance (Kusiak and Chung, 1991). The choice of vigilance parameter is application specific, and is usually determined by experimentation. In some applications a variable vigilance parameter has also been used.

2.3. *Learning*

Learning updates top-down and bottom-up weights until equilibrium weights are obtained. Once a cluster has been selected for learning, the bottom-up and top-down signals are maintained for an extended period, during which weight changes occur. This is the 'resonance' that gives the theory its name. Learning can occur in two modes: fast learning and slow learning. In the fast learning mode, weight updates during resonance occur rapidly, whereas in slow learning mode the changes are slow, relative to the length of time a pattern is presented on any particular trial. A learning trial in ART consists of the presentation of one input pattern. Many more presentations of the patterns are required for slow learning than for fast learning. A fast learning mode is typically used for ART1, whereas a slow learning mode appears to be more suitable for ART2 (Fausett, 1994). Slow learning is less susceptible to noise, and is not influenced by the order of presentation of the patterns.

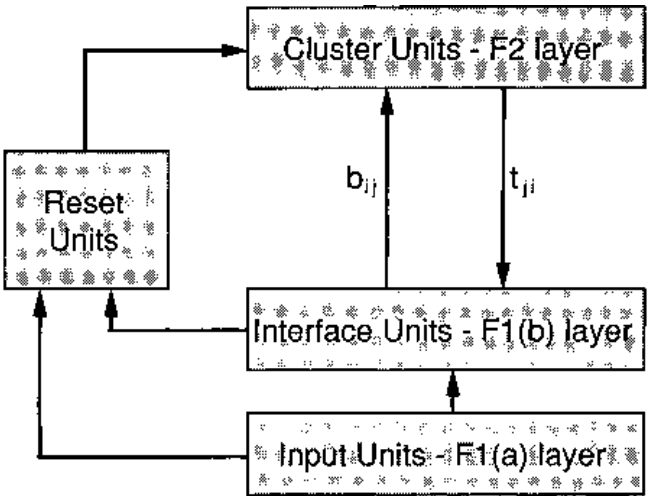

Fig. 1. Basic structure of ART networks.

2.4. *ART2 versus ART1*

The differences between ART2 and ART1 reflect the modifications needed to accommodate patterns with continuous-valued components. The F1 field of ART2 is more complex because continuous-valued input vectors may be arbitrarily close together. The F1 field in ART2 includes a combination of normalization and noise suppression, in addition to the comparison of the bottom-up and top-down signals needed for the reset mechanism. A typical ART2 architecture is shown in Fig. 4 and an algorithm is explained step by step in Section 4.

3. Face score vectors

Hwang and Henderson (1992) proposed the concept of face score vectors in order to represent features as vectors suitable for neural network input. Face score is a function of the face, edge and vertex geometries, and is a measure of the face complexity based upon the convexity or the concavity of the surrounding region. The face score vector depicts the topological relations between a face and its neighbouring faces. A modified face score calculation and vector formation method is used in this paper. Each face of

a solid, represented by a B-rep model, has a number of edges and vertices. Also, each face may have one or more inner loops. The following scores are assigned to edges, loops and face geometry:

Edge scores (E)	
Convex edge	+0.5
Concave edge	−0.5
Loop scores (L)	
Positive inner-loop	+1.0
Negative inner-loop	−1.0
Face geometry scores (F_g)	
Plane surface	0
Convex surface	+2.0
Concave surface	−2.0

The numerical values chosen for the edge, loop and face geometry scores reflect the geometric nature of the entities. Face geometry is more important than the inner loops on the face, which, in turn, are more important than the edges in determining a feature. By using positive and negative scores for geometrically opposite entities, a strikingly different representation is obtained for geometrically opposite features such as a slot and a tab.

The vertex score is calculated by:

$$V = \sum_{i=1}^{3} E_i \tag{1}$$

where V is the vertex score, and E_1, E_2 and E_3 are the scores of the three edges that intersect to form the vertex.
The face score is given by:

$$F = \sum_{j=1}^{n} \frac{V_j}{n} + F_g + \sum_{k=1}^{m} L_k \tag{2}$$

where n is the number of vertices shared by the face, and m is the number of inner loops present on the face.

A slot feature is shown in Fig. 2(a). The ten faces are identified by numbers 1–10 inscribed at the center of the corresponding face. In Fig. 2(b), a two-dimensional representation of the solid is given, depicting the adjacency relationships between the faces. The face score calculation for each of the ten faces is illustrated below.

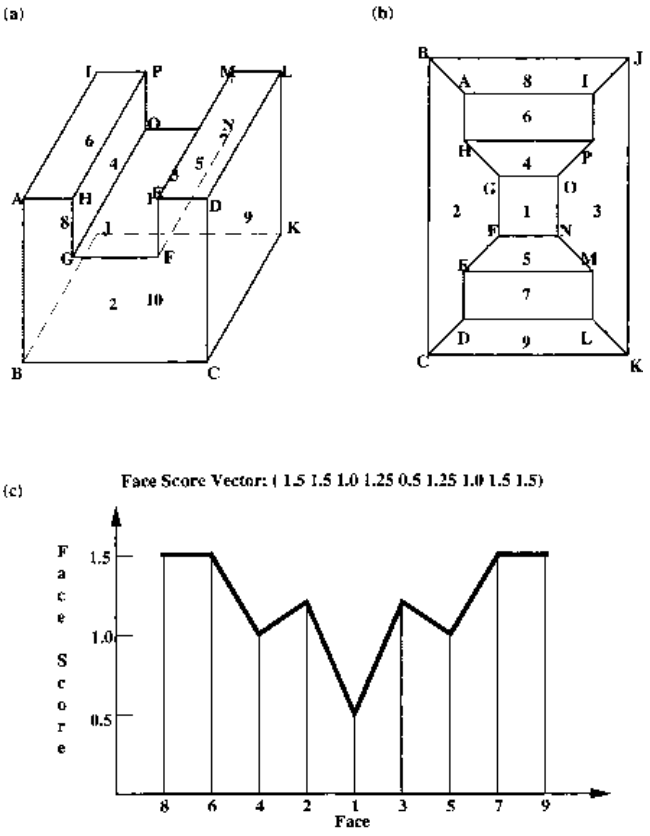

Fig. 2. (a) A slot feature; (b) two-dimensional representation of the solid, depicting the adjacency relationships of the faces; (c) graphical representation of the same vector.

3.1. *Face score calculation*

In Fig. 2(a), all the edges are convex except GO and FN. Therefore all the vertices except G, F, O and N have a vertex score equal to 1.5, whereas vertices G, F, O and N have a vertex score of 0.5 each according to Equations 1 and 2. Because all the faces are planar, the face geometry score corresponding to each face is equal to zero. There are no inner loops associated with any face in this example. The face score values corresponding to each face are listed in Table 1.

Table 1. Face scores for the slot feature

Face no.	*Face (s)*	*Face score*
1	GFNO	$\frac{0.5+0.5+0.5+0.5}{4} = 0.5$
2, 3	ABCDEFGH IJKLMNOP	$\frac{1.5+1.5+1.5+1.5+1.5+0.5+0.5+1.5+1.5}{8} = 1.25$
4, 5	GHPO and FEMN	$\frac{0.5+1.5+1.5+0.5}{4} = 1.0$
6–10	AHPI, EMLD, ABJI, DCKL, BCKJ	$\frac{1.5+1.5+1.5+1.5}{4} = 1.5$

3.2. *Face score vector formation*

A nine-element face score vector corresponding to each face is formed in accordance with the following rules:

(1) The fifth element of the vector is the face score of the face under consideration (main face);

(2) The immediately adjacent faces with highest scores comprise the fourth and the sixth elements, the next highest scores are the third and the seventh elements, and so on;

(3) If the solid has less than nine faces, then the remaining elements of the vector are set to 1.5.

A graphical representation of the face score vector corresponding to face 1 of the slot feature is shown in Fig. 2(c). The fifth element of the vector has a score of 0.5 corresponding to the score of face 1. Faces 2, 3, 4 and 5 are immediately adjacent to face 1. Of the faces 2, 3, 4 and 5, 2 and 3 have a higher score than 4 and 5: hence they comprise the fourth and sixth elements of the face score vector, and faces 4 and 5 comprise the third and seventh elements. Faces 6, 7, 8 and 9 comprise elements 2, 8, 1 and 9 respectively. The face score of the tenth face does not appear in the face score vector corresponding to face 1.

Because the effect of the faces far away from the main face plays a minor role in determining the feature associated with the main face, a nine-element vector is deemed sufficient for this purpose. The face score vectors for nine intermediate features, which are input to the net for clustering, are shown in Fig. 3.

4. ART2 algorithm

The following ART2 algorithm is taken from Fausett (1994). A typical ART2 architecture (Fausett, 1994) is reproduced in Fig. 4 for reference. Notice the additional layers present in the F1 layer compared with the basic architecture presented in Fig. 1. The computational cycle of the F1 layer can be initiated with the calculation of the activation (output) of the layer U. Each of the layers P, Q, U, V, W, X and R has the same number of units, equal to the number of elements in the input vector (nine in the present case).

Before proceeding with the algorithm, two functions, normalize and threshold, need to be defined. In the following, bold italic letters indicate vectors.

Let $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$

$$N(\boldsymbol{x}) = \text{Normalize } (\boldsymbol{x}) = \frac{\boldsymbol{x}}{\|\boldsymbol{x}\|}$$

where $\|\boldsymbol{x}\| = \sqrt{(x_1^2 + x_2^2 + \cdots + x_n^2)}$

$$T(\boldsymbol{x}) = \text{Threshold } (\boldsymbol{x}) = (T(x_1), T(x_2), \ldots T(x_n))$$

$$T(x_i)\text{Threshold } (x_i) = \begin{cases} x_i & \text{if } x_i \geqslant \theta \\ 0 & \text{if } x_i < \theta \end{cases}$$

The algorithm consists of the following steps.

Step 0

Initialize parameters
$a = 10$; $b = 10$; $\theta = 0.1$; $c = 0.1$; $d = 0.9$;
α(learning rate) $= 0.6$; ρ (vigilance parameter) $= 0.9$;
for $i = 1$ to n (number of nodes in F1 layer) and
$j = 1$ to m (number of clusters)
$t_{ji} = 0$ and
$b_{ij} \leqslant \frac{1}{(1-d)\sqrt{n}}$

Step 1

Do steps 2–12 N_EP (number of epochs) times. An epoch is one presentation of each pattern.

Step 2

Do steps 3–11 for each input vector $\boldsymbol{s}$.

Step 3

$\boldsymbol{u} = \boldsymbol{0}$, $\boldsymbol{w} = \boldsymbol{s}$, $\boldsymbol{p} = \boldsymbol{0}$, $\boldsymbol{x} = N(\boldsymbol{s})$, $\boldsymbol{q} = \boldsymbol{0}$, $\boldsymbol{v} = T(\boldsymbol{x})$;
$\boldsymbol{u} = N(\boldsymbol{v})$, $\boldsymbol{w} = \boldsymbol{s} + a\boldsymbol{u}$, $\boldsymbol{p} = \boldsymbol{u}$, $\boldsymbol{x} = N(\boldsymbol{w})$, $\boldsymbol{q} = N(\boldsymbol{p})$,
$\boldsymbol{v} = T(\boldsymbol{x}) + bT(\boldsymbol{q})$.

Step 4

Compute the net input of the F2 units.
For $j = 1$ to m

$$y_j = \sum_{i=1}^{n} b_{ij} p_i$$

Step 5

While reset is true, do steps 6–7.

Step 6

Find $J = j$, corresponding to the F2 unit with largest net input.

Step 7

Check for reset.
$\boldsymbol{u} = N(\boldsymbol{v})$,
for $i = 1$ to n

$$p_i = u_i + dt_x$$

$$r_i = \frac{u_i + cp_i}{\|\boldsymbol{u}\| + c\|\boldsymbol{p}\|}$$

If $\|\boldsymbol{r}\| < \rho$, then
$y_J = -1$ (inhibit Jth node from participation)
reset = true
repeat step 5

If $\|\boldsymbol{r}\| \geqslant \rho$, then
$\boldsymbol{w} = \boldsymbol{s} + a\boldsymbol{u}$, $\boldsymbol{x} = N(\boldsymbol{w})$, $\boldsymbol{q} = N(\boldsymbol{p})$, $\boldsymbol{v} = T(\boldsymbol{x}) + bT(\boldsymbol{q})$
reset = false
go to step 8

Step 8

Do steps 9–11 N_IT times (N_IT: number of iteration, equal to 1 in the present case).

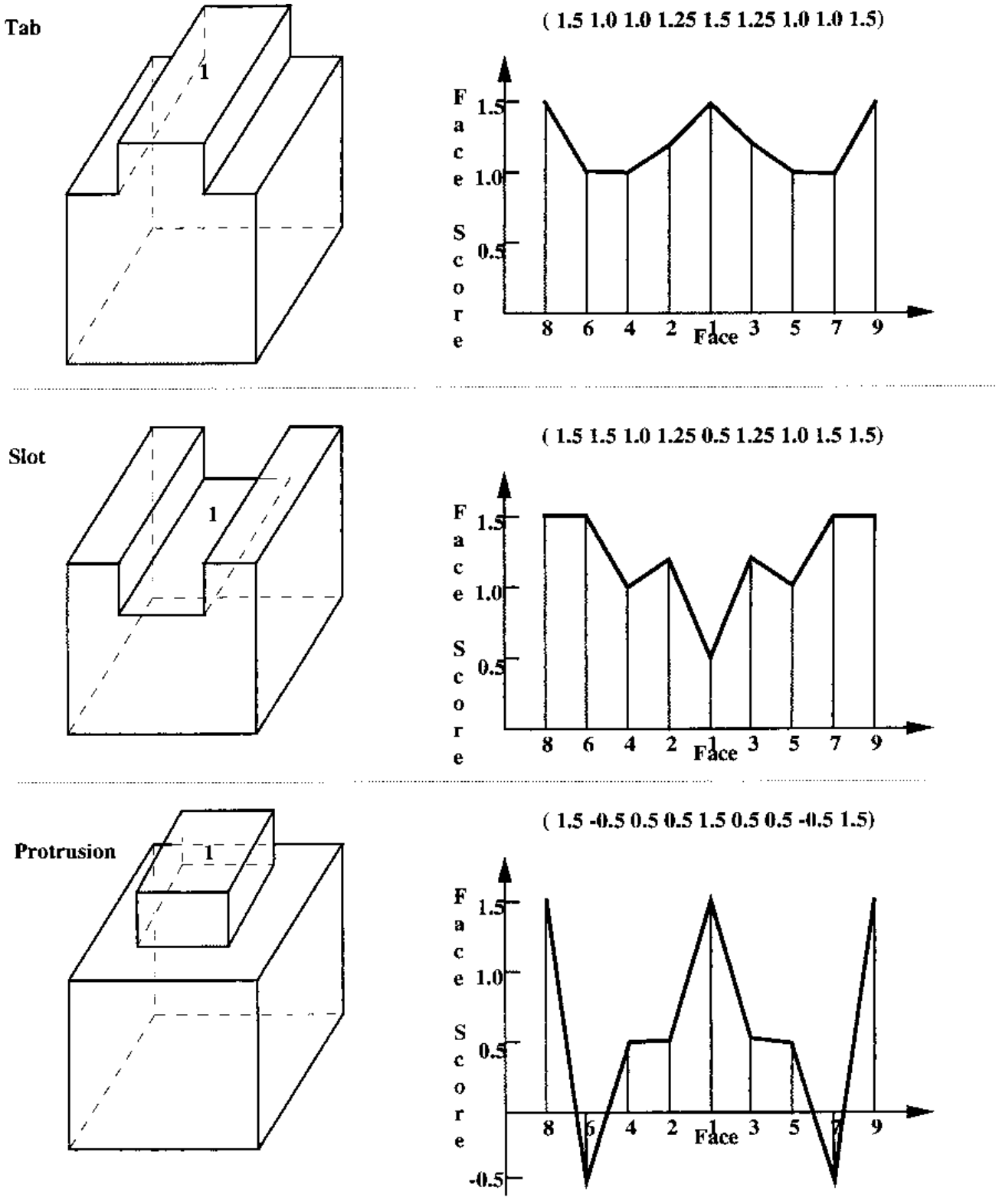

Fig. 3. Face score vectors of nine features.

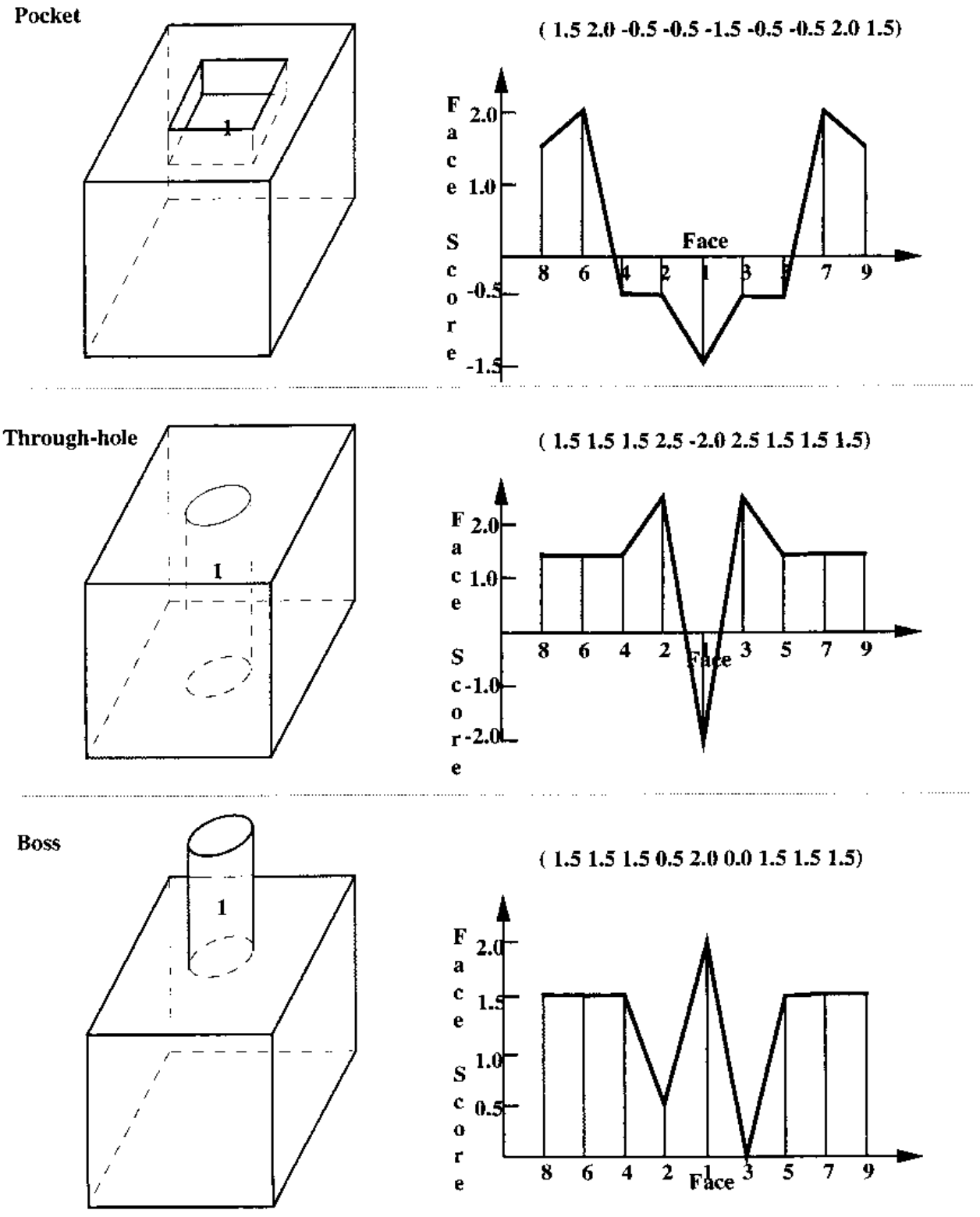
Pocket
(1.5 2.0 -0.5 -0.5 -1.5 -0.5 -0.5 2.0 1.5)
Face Score
2.0
1.0
-0.5
-1.5
Face
8 6 4 2 1 3 5 7 9
Through-hole
(1.5 1.5 1.5 2.5 -2.0 2.5 1.5 1.5 1.5)
Face Score
2.0
1.0
-1.0
-2.0
Face
8 6 4 2 1 3 5 7 9
Boss
(1.5 1.5 1.5 0.5 2.0 0.0 1.5 1.5 1.5)
Face Score
2.0
1.5
1.0
0.5
Face
8 6 4 2 1 3 5 7 9

Fig. 3. (*contd.*)

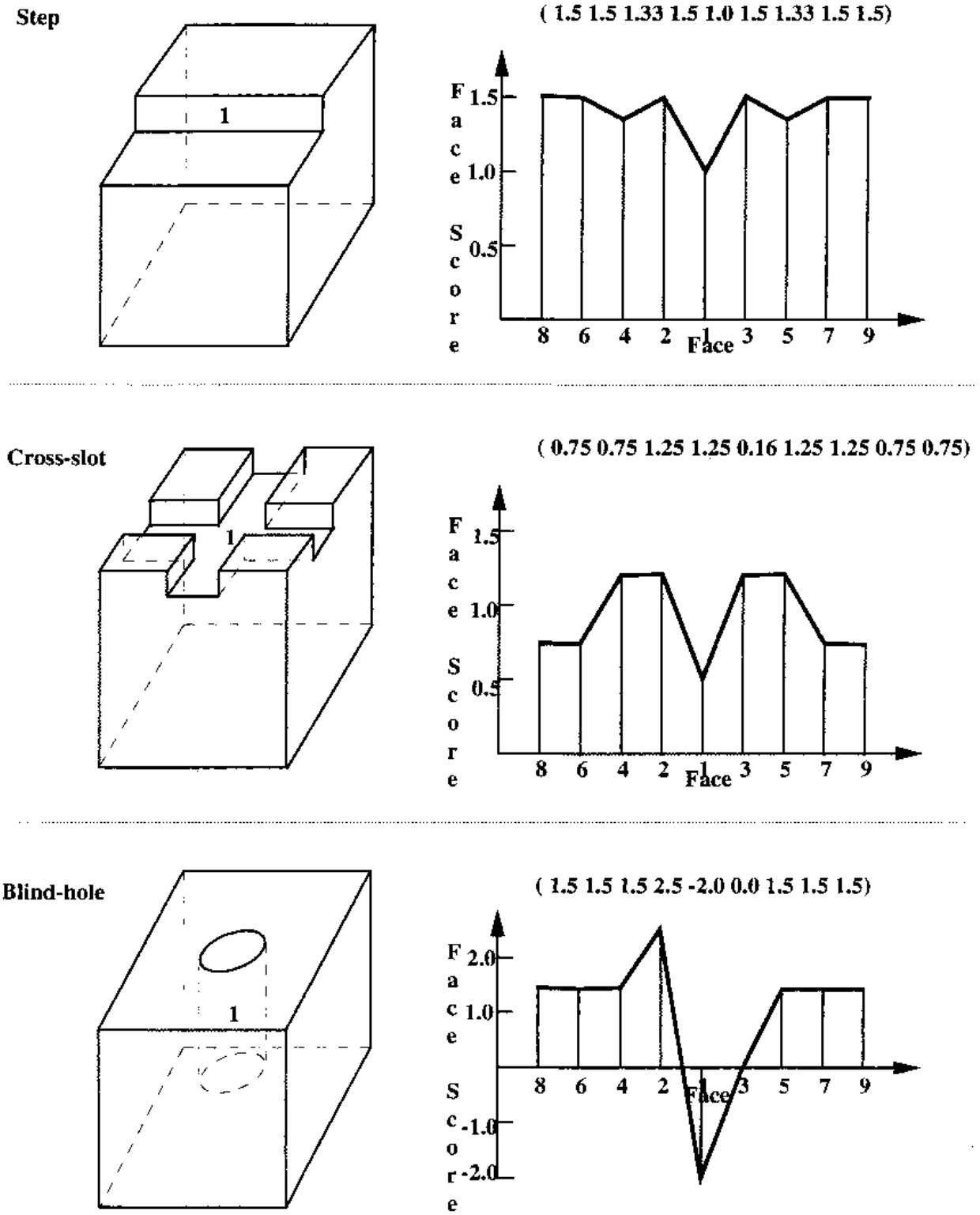
Step
(1.5 1.5 1.33 1.5 1.0 1.5 1.33 1.5 1.5)
1
Face Score
1.5
1.0
0.5
8 6 4 2 1 3 5 7 9
Face
Cross-slot
(0.75 0.75 1.25 1.25 0.16 1.25 1.25 0.75 0.75)
1
Face Score
1.5
1.0
0.5
8 6 4 2 1 3 5 7 9
Face
Blind-hole
(1.5 1.5 1.5 2.5 -2.0 0.0 1.5 1.5 1.5)
1
Face Score
2.0
1.0
-1.0
-2.0
8 6 4 2 1 3 5 7 9
Face

Fig. 3. (*contd.*)

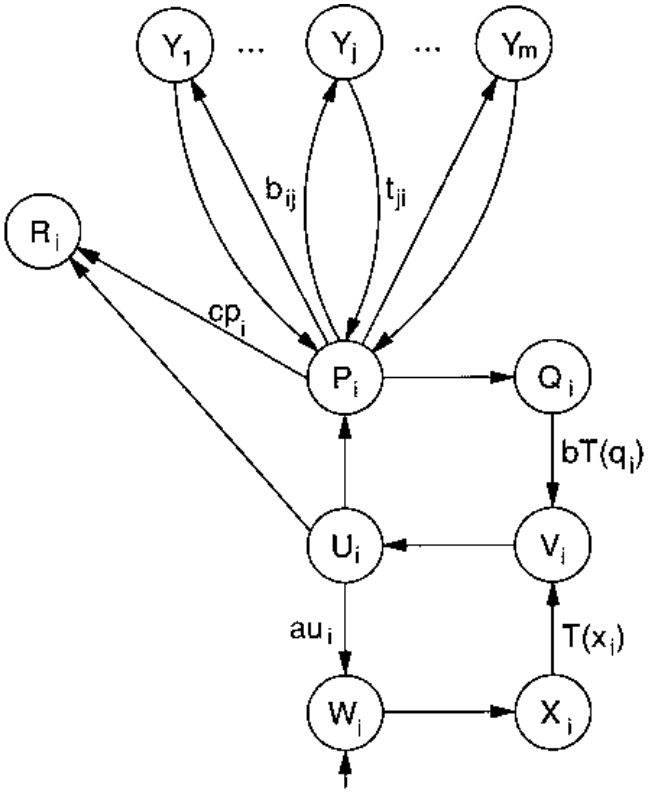

Fig. 4. A typical ART2 architecture.

Step 9

Update weights for unit J.

$$t_{Ji} = \alpha d u_i + \{1 + \alpha d(d-1)\} t_{Ji}$$
$$b_{iJ} = \alpha d u_i + \{1 + \alpha d(d-1)\} b_{iJ}$$

Step 10

$$\boldsymbol{u} = N(\boldsymbol{v}), \quad \boldsymbol{w} = \boldsymbol{s} + a\boldsymbol{u}, \quad \boldsymbol{p} = \boldsymbol{u} + d\boldsymbol{t}_J,$$
$$\boldsymbol{x} = N(\boldsymbol{w}), \quad \boldsymbol{q} = N(\boldsymbol{p}), \quad \boldsymbol{v} = T(\boldsymbol{x}) + bT(\boldsymbol{q}).$$

Step 11

Test stopping condition for weight updates.

Step 12

Test stopping condition for number of epochs.

5. Results and discussion

The algorithm for the ART2 network, presented in the previous section, was implemented in MATLAB^{TM}. The face score vectors corresponding to the nine features presented in Section 3 are input to the ART2 network. The net clustered the nine patterns into eight different clusters, as shown in Table 2. A value of 0.9 is used for the vigilance parameter, and slow learning mode is used for updating weights (N_IT = 1). The noise suppression parameter (θ) is chosen to be 0.1. In Table 2, the first column corresponds to the features that are being clustered, and they are input to the net from tab to blind hole in the order given. In the second column the output node to which the feature is clustered after each epoch is shown. An epoch is one presentation of each pattern to the net.

Table 2. Network results

Feature	*Cluster*		
	Epoch 1	*Epoch* 2	*Epoch* 3
Tab	1	6	6
Slot	2	7	7
Protrusion	1	1	1
Pocket	2	2	2
Through-hole	3	3	3
Boss	4	4	4
Step	5	8	8
Cross-slot	5	5	5
Blind-hole	3	3	3

When the face score vector corresponding to the tab feature was input to the net for the first time it was associated with the first output node. Then the slot feature formed a new cluster. The next two features, protrusion and pocket, were clustered by the first and second nodes respectively. This indicates that the weights corresponding to the first two nodes have not changed significantly after two presentations to differentiate between tab and protrusion and slot and pocket. The next three features created their own clusters. Cross-slot was grouped with step and blind hole with through-hole.

By the end of the second epoch, the network learned to differentiate between tab and protrusion, slot and pocket and step and cross-slot. But the network still clustered through-hole and blind hole together. By the end of the second epoch the net stabilized as it repeated the same classification in the ensuing epochs. By increasing the number of epochs, equilibrium weights were obtained. Figures 5 and 6 show graphical representations of the top-down and bottom-up weights after 10 and 50 epochs respectively. Initially, the top-down weights (t_{ji}) are set to zero, and bottom-up weights (b_{ji}) are set to a constant value, as given in the algorithm. As shown in Fig. 5, by the end of ten epochs the weights change slightly towards the shape of the input pattern. Figure 6 shows that the weights for all the clusters except cluster 3 stabilize to values similar to the input patterns. Because cluster 3 has learned both through- and blind holes, the corresponding weights stabilized to a pattern intermediate to the two patterns.

The effect of the changes in the vigilance and the noise suppression parameters on the network performance were investigated, and the results are summarized in Table 3. The cluster units shown in Table 3 correspond to stabilized clustering at the chosen values of θ and ρ. The network did not differentiate between through-hole and blind hole until

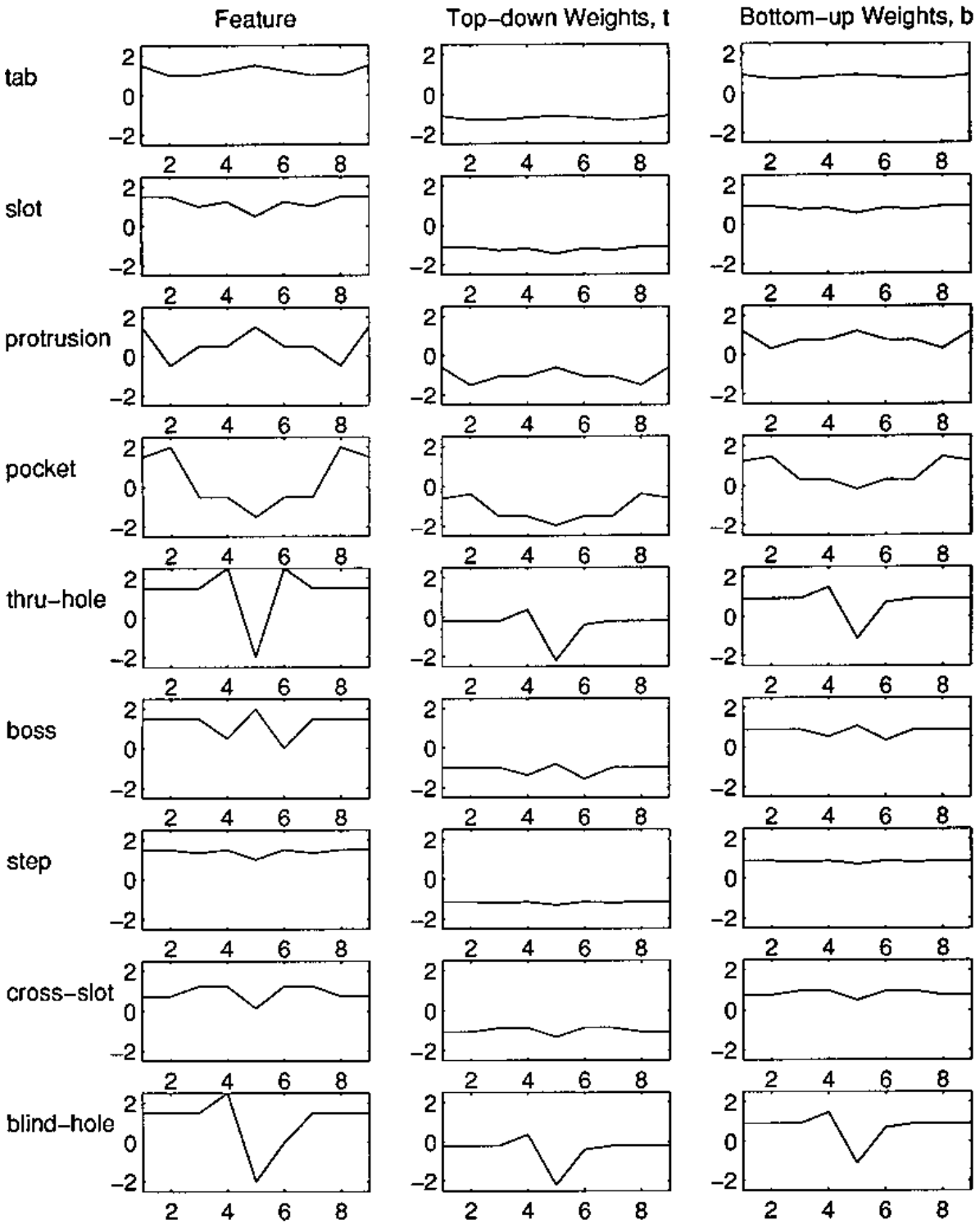

Fig. 5. The status of the top-down and bottom-up weights after ten epochs for the nine features.

the vigilance parameter (ρ) was increased to 0.9995. The values of the noise suppression parameter (θ) between 0.1 and 0.3 did not affect the net performance. At a value of θ equal to 0.3 and ρ equal to 0.9 the clustering became coarse, and the net clusters slot, through-hole, step, cross-slot and blind hole features together. As the vigilance parameter was increased the clustering became finer. When ρ was increased to 0.995 the network formed a separate cluster for blind hole, and as ρ was further increased to 0.9999, the slot and step features were classified together

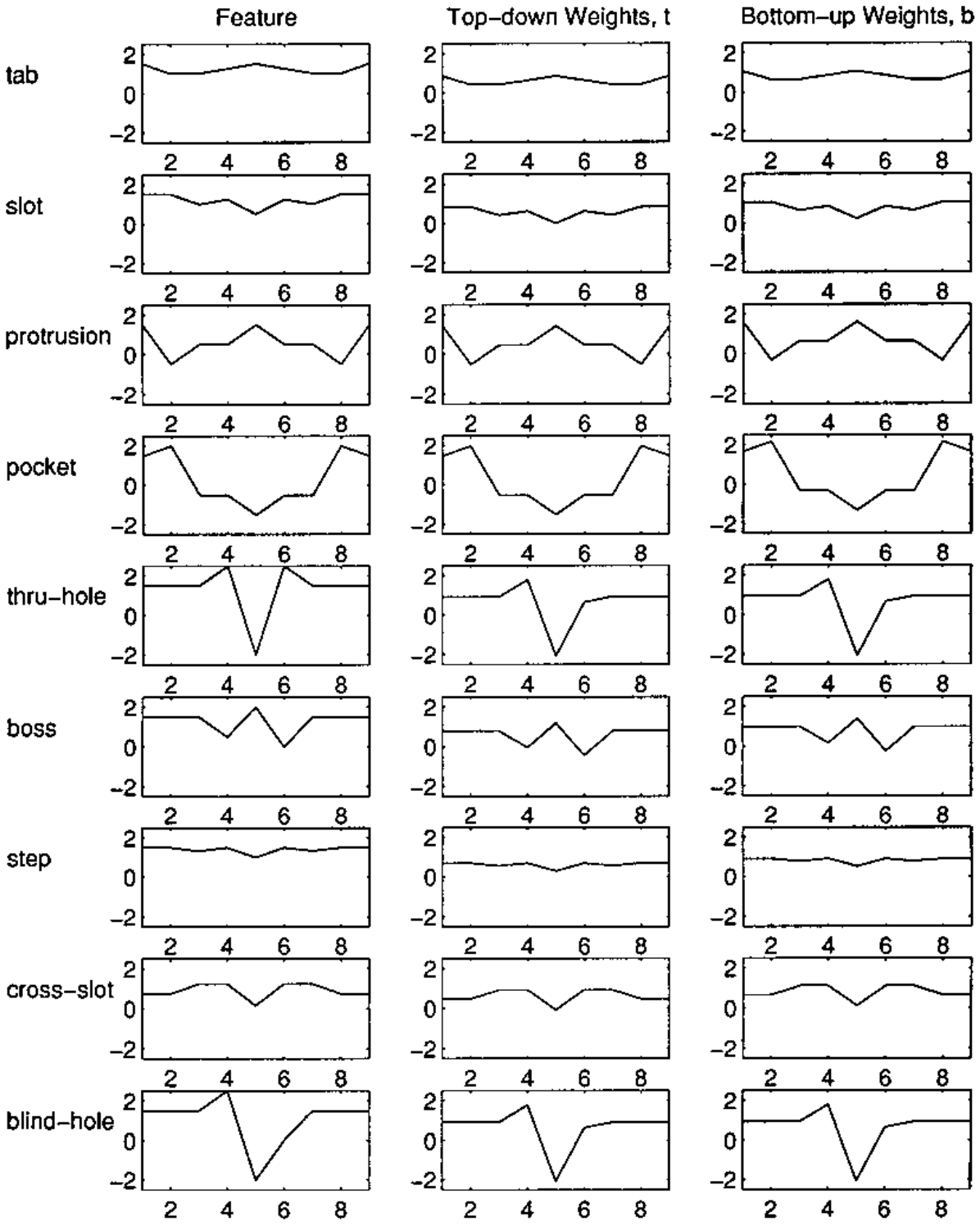

Fig. 6. The status of the top-down and bottom-up weights after 50 epochs for the nine features.

with the other features forming individual clusters. Finally, when ρ was made equal to 0.999 99, nine different clusters were formed. The components of the normalized input vector that are less than the noise suppression ratio, θ, were set to zero. When θ was increased to 0.3, some of the values in the input vectors were set to zero, and this made some of the patterns more similar to each other. This explains the coarser clustering that occurred when θ was set to 0.3 and ρ was set to 0.9. The values of ρ below 0.9 were observed to have no effect on the clustering.

Table 3. Sensitivity analysis

Feature	*Cluster units*					
	$\theta = 0.1$		$\theta = 0.3$			
	$\rho = 0.9$	$\rho = 0.9995$	$\rho = 0.9$	$\rho = 0.995$	$\rho = 0.9999$	$\rho = 0.999\,99$
Tab	6	7	5	5	1	1
Slot	7	2	3	3	2	2
Protrusion	1	1	1	1	3	3
Pocket	2	3	2	2	4	4
Through-hole	3	4	3	3	8	5
Boss	4	5	4	4	5	6
Step	8	8	3	3	2	9
Cross-slot	5	6	3	3	6	7
Blind hole	3	9	3	6	7	8

The results obtained in this paper only indicate the potential of using unsupervised ART neural networks for feature recognition from a solid model. The main advantages of the neural network approach are high recognition speed, ability to recognize partial and complex features, minimal memory storage, and ease of computation. Because both graph-based and rule-based systems perform exhaustive searches for matching their patterns, the execution time can become very high, depending on the complexity of the component. Neither graph-based nor rule-based systems have the ability to learn or dynamically improve their performance. The performance of the net depends highly on the representation of the features, and further research is needed to develop a more robust face score vector calculation scheme or some other way of representing the features. The possibility of applying some other variations of ART networks, such as fuzzy ART, for feature recognition problem should be considered in the future.

References

Carpenter, G. A. (1989) Neural network models for pattern recognition and associative memory. *Neural Networks*, **2**, 243–257.

Carpenter, G. A. and Grossberg, S. (1987a) A massively parallel architecture for a self-organizing neural pattern recognition machine. *Computer Vision, Graphics, and Image Processing*, **37**, 54–115.

Carpenter, G. A. and Grossberg, S. (1987b) ART2: self-organization of stable category recognition codes for analog input patterns. *Applied Optics*, **26**, 4919–4930.

Chang, T. C. (1990) *Expert Process Planning for Manufacturing*, Addison-Wesley, Reading, MA, pp. 73–103.

Choi, B. K. (1982) CAD/CAM compatible tool oriented process planning system, PhD Thesis, Purdue University.

Chuang, S. (1991) Feature recognition from solid models using conceptual shape graphs, PhD Thesis, Arizona State University.

Fausett, L. (1994) *Fundamentals of Neural Networks: Architectures, Algorithms and Applications*, Prentice-Hall, Englewood Cliffs, NJ, pp. 218–287.

Gadh, R. and Prinz, F. B. (1991) Shape feature abstraction in knowledge-based analysis of manufactured products, in *Proceedings of the 7th Conference on AI Applications*, Miami Beach, FL, IEEE Computer Society Press, CA, pp. 198–204.

Henderson, M. R. (1984) Extraction of feature information from three dimensional CAD data, PhD Thesis, Purdue University.

Hwang, J. (1988) Rule-based feature recognition: concepts, primitives and implementation, MS Thesis, Arizona State University.

Hwang, J. -L. and Henderson, M. R. (1992) Applying the perceptron to three-dimensional feature recognition. *Journal of Design and Manufacturing*, **2**(4), 187–198.

Joshi, S. and Chang, T. C. (1988) Graph-based heuristics for recognition of machined features from a 3D solid model. *Computer Aided Design*, March, **20**(2), pp. 58–66.

Kung, H. (1984) An investigation into development of process plans from solid geometric modeling representation, PhD Thesis, Oklahoma State University.

Kusiak, A. and Chung, Y. (1991) GT/ART: using neural networks to form machine cells. *Manufacturing Review*, **4**(4), 293–301.

Kyprianou, L. K. (1980) Shape classification in computer aided design, PhD Thesis, University of Cambridge, UK.

Sakurai, H. and Gossard, D. C. (1988) Shape feature recognition from 3D solid models, in *Proceedings of the 1988 ASME International Computers in Engineering Conference*, American Society of Mechanical Engineers, NY, pp. 515–519.

Vanderbrande, J. (1990) Automatic recognition of machinable features in solid models, PhD Thesis, University of Rochester.