3

A New Self-Organizing Neural Network Architecture for Parallel Multimap Pattern Recognition—FAUST

C.L. Wilson
Advanced Systems Division
National Institute of Standards and Technology
Gaithersburg, MD

1. INTRODUCTION

Previous work has demonstrated that it is possible to use adaptive resonance methods [1,2,3], such as ART-1 [4], for feature detection in image recognition problems if the images involved have been appropriately preprocessed. In the CORT-X method [1], these filters are formed to approximate known neural sensitivity patterns. (In the neocognitron [5], like method [2], the image is segmented into regional features, and in [3] Gabor filters [6] are used to approximate neural receptor profiles.)

All of these methods require multiple layers of neural processors and include a priori assumptions about the nature of the filtering or segmentation required for the pattern recognition problem. The addition of layers of processors decreases recognition speed by lowering the degree of parallelism in the system. A priori assumptions can cause the system to be specialized to a narrower range of applications and decreases system flexibility.

1.1. Generalized Self-Organizing Structure

The generalized structure of a self-organizing pattern recognition system is shown in Figure 3.1. This structure is sufficiently general to provide a model for ART-1, ART-2, CORT-X, and FAUST. In the ART-1, the filter does not couple to the memory or to the learning control. In ART-2, the filter is tightly coupled to the memory and to the learning control. In CORT-X, an additional layer of

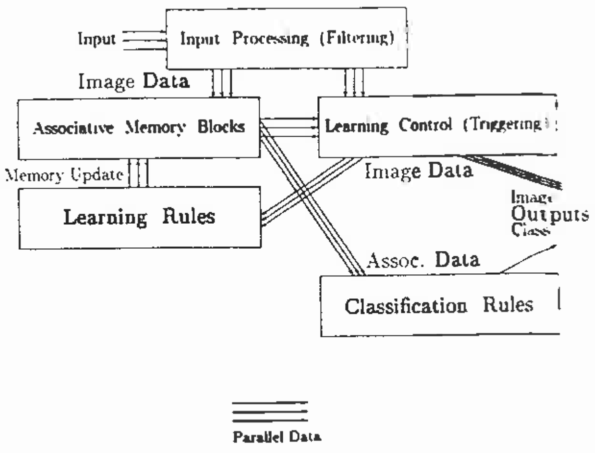

Figure 3.1. General architecture for self-organizing pattern recognition system.

filtering is placed in front of the ART-2 filter. In FAUST, the filter is independent of memory and learning control, but drives both.

Neglecting the method of initialization until Section 3, all four methods work by taking in input images, reconstructing them in the filter, and passing the image to both the associative memories and the learning control. The associative memories compare the filtered image to previously stored images using association rules. The associative memory structure contains short-term memory, STM, and long-term memory, LTM, blocks for ART-1, ART-2, and CORT-X. The FAUST associative memory structure can contain any number of blocks; in the example presented here, two blocks are used: (1) pattern memory, and (2) relevance memory. The associative memories generate two association strengths, one for each block, for each memory location.

The association strengths are used to gate image data from the filter to the learning section. In the ART-based methods, resonance is used. Faust uses a method called association space mapping, which is explained in Section 4. The image data which passes through the learning control is used in the learning block to store new data into the associative memory location which triggers learning. In FAUST, six different learning rules have been used. Classification of patterns can be either by Bayesian statistics or by association space mapping.

Any recognition system must provide:

1. scale invariance,
2. rotational invariance,
3. edge position invariance, and
4. local shape invariance.

These four requirements are difficult to achieve at reasonable cost on serial machines. In [3] it was demonstrated that Items 3 and 4 could be implemented effectively on a massively parallel computer using Gabor filtering. In the present work, Items 1 and 2 are implemented using massive parallelism.

The principal contrast between the present work and [1,3,4] is that the partitioning of tasks between the filter section and the learned associations is different. In [1] position invariance is achieved in the filter section and shape invariance is learned in a closely coupled nonlinear filter. In the present work, shape invariance is achieved through filtering and edge position invariance is learned.

1.2. FAUST Architecture

Previous work by Linsker [7] and Rubner [8] has shown that feed-forward learning rules are capable of generating layers of neural processors that have sensitivity profiles similar to the sensitivity profiles found in the visual system and the Gabor basis functions [6] used in [3]. This leads to the query: How can these learning processes be combined directly into a self-organizing system which allows the filter sensitivity to be learned in parallel with pattern recognition features?

This chapter discusses one possible answer to this query. The answer proposed here is that this can be achieved using a multimap procedure similar to those known to exist in the mid-level visual cortex [9]. To be equivalent to previous work [1,2,3] the method also must provide a parallel, multimap, self-organizing, pattern classification procedure. In this chapter, this problem is solved using a feed-forward architecture which allows multimap features stored in associative memories to be accessed in parallel and to trigger a symmetrically controlled parallel learning process. This method allows features of different data type, such as binary image patterns and eight-bit statistical correlations, to be updated in parallel. This parallel updating process provides feature flexibility similar to [10], but uses a resonance procedure [4] to initiate learning. The architecture is described by the acronym FAUST (Feed-forward Association Using Symmetrical Triggering). A diagram of the FAUST architecture is shown in Figure 3.2. The three essential features of FAUST, shown in this figure, are: (1) Different feature classes use individual association rules, (2) Different feature classes use individual learning rules, and (3) All feature classes contribute symmetrically to learning. The number of feature classes is shown as two in Figure 3.2 for graphic clarity, but the architecture is not restricted to any number or type of feature classes. The upper summation over the pattern memory in Figure 3.2 is carried out using functions of the type given for binary data in Section 2 to produce a pattern association strength, P. The lower summation over feature relavance memory in Figure 3.2 is carried out using functions of the type given

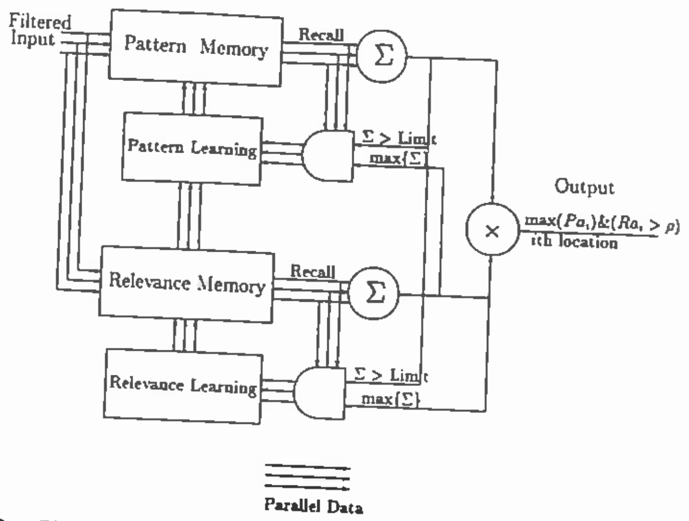

Figure 3.2. FAUST architecture as implemented on the parallel processor array.

for eight-bit data in Section 2 to produce a relevance association strength, R. The "Limit" shown on the two gates in Figure 3.2 is the learning threshold, ρ.

The FAUST method differs from [9] in that in [9] a hierarchy of maps is created using supervised learning. In FAUST, the type of features to be learned are contained in separate memory blocks similar to the "patches" used in [10]. All learning in FAUST is self-organizing and no preclassification of data is used during learning. In the character-recognition example presented here, five different association rules and four different learning rules are used to demonstrate that the recognition performance of FAUST is not sensitive either to the association or to the learning rules of the method.

FAUST is a single layer, laterally parallel, resonance method. It divides associative memory into associative memory blocks each of which represents a single feature class or type which constitutes a class of memory maps. In imaging applications the memory locations contain learned images, which are processed in parallel-by-map type. FAUST differs from the ART resonance methods in that all of the associative memory blocks are connected directly to the input and all associative memories contribute to and are connected to, through the trigger circuit, the output. This illustrates the critical role of the Symmetrical Triggering, ST, part of FAUST in the operation of the architecture. Experiments presented here will show that the operation of FAUST is weakly dependent on both the association and learning rules. Then how is it different from an associative memory?

FAUST gains its pattern recognition capability from its self-organized learning capability, which is initiated by the ST mechanism. The symmetrical part comes from the use of identical triggering rules on every learning gate. These gates maintain a fixed relationship between the feature types. This relationship is also what maintains the linear character of the method and eliminates the need for a double parameter search over all combinations of feature classes. This symmetry derives from a logical relationship between the input and the information retained in the associative memories, so that each address comprises an input related, self-organized, feature set.

In the FAUST architecture, the learning trigger logic takes place in an $n \times m$-dimensional space when n-pattern and m-relevance feature classes are used. In ART architecture the equivalent learning-logic space is one-dimensional and was extended for parallel computation in [3] to two dimensions. The n-binary classes can be regarded as existence classes or vigilance classes which allow, but do not select learning. The m-multibit or analog-relevance classes cannot allow learning, but select what is learned when learning is allowed by the existence features.

1.3. Outline of Chapter

The organization of the chapter follows the parallel data flow shown in Figure 3.2. Incoming data is filtered and presented, in parallel, to each of the associative memory maps. The filtering methods used are discussed in Section 2. The methods of associative recall used to locate the best match are discussed in Section 3. After the association strength has been determined for each memory location of each memory map, learning is triggered by a set of parallel comparisons that gate input-image data to the learning modules. This symmetric triggering of learning is discussed in Section 4. When learning has been triggered, the learning module for each memory map updates the selected memory locations. The learning methods used are discussed in Section 5. Once the associative learning has progressed to a point where a sufficient number of patterns are retained in memory, pattern classification takes place using the same logic used in symmetric triggering to access the classification history of the learned patterns. Pattern classification is discussed in Section 6. The stability of the system which results from these steps is discussed in Section 7. As an example of the ability of FAUST to perform self-organizing pattern recognition, a character-recognition example is presented in Section 8.

2. IMAGE FILTERING

The input filtering module consists of eight sections, up to four of which can be connected in cascade to provide a multilevel filtering capability. The signal flow through the composite filter is in the order shown in Table 3.1. The data flow through each filter is shown in Figure 3.3. The specific operations performed by each functional module are discussed in order below.

Table 3.1. Table of the Possible Filter Types Used in the Cascade Filter Module

Function	Parameters	Exclusions
Normalize	32×32	none
Shear	internal	none
Walsh-8	8×8 basis	all other Walsh
Walsh-10	10×10 basis	all other Walsh
Walsh-13	13×13 basis	all other Walsh
Walsh-16	16×16 basis	all other Walsh
Gabor-16	16 even functions	other Gabor
Gabor-32	32 even-odd funct.	other Gabor

2.1. Normalization

Normalization is used to provide limited scale invariance to each character image. Each image initially is represented by a picture area greater than 32 × 32 pixels. In the normalization process the active image area is scaled so that the largest dimension of the image is 32 pixels and centered so that the image has equal numbers of empty pixels on either side of the image in the other dimension. This process either replicates pixels to enlarge the image or it deletes pixels to reduce image size. The effects of the process on machine-printed characters is shown in Figure 3.4. The change in the "*A*" in the upper left is small. The "*g*," right of the "*A*," is raised to move the distender into the 32 × 32 box. The ".," right of the "*g*," is magnified to resemble a crude-filled circle. The aspect ratio of the image is maintained as accurately as possible. Normalization can destroy the differences between characters; as an example the "*O*" and "*o*" in Figure 3.4 are no longer different. Typical normalization time is 730μ sec. per character.

2.2. Shear Transform

The shear transform maps a parallelogram-shaped region of the image into a centered rectangular region of the image. The amount of shear is determined by

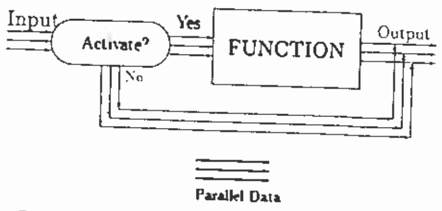

Figure 3.3. Parallel data flow through a single filter module.

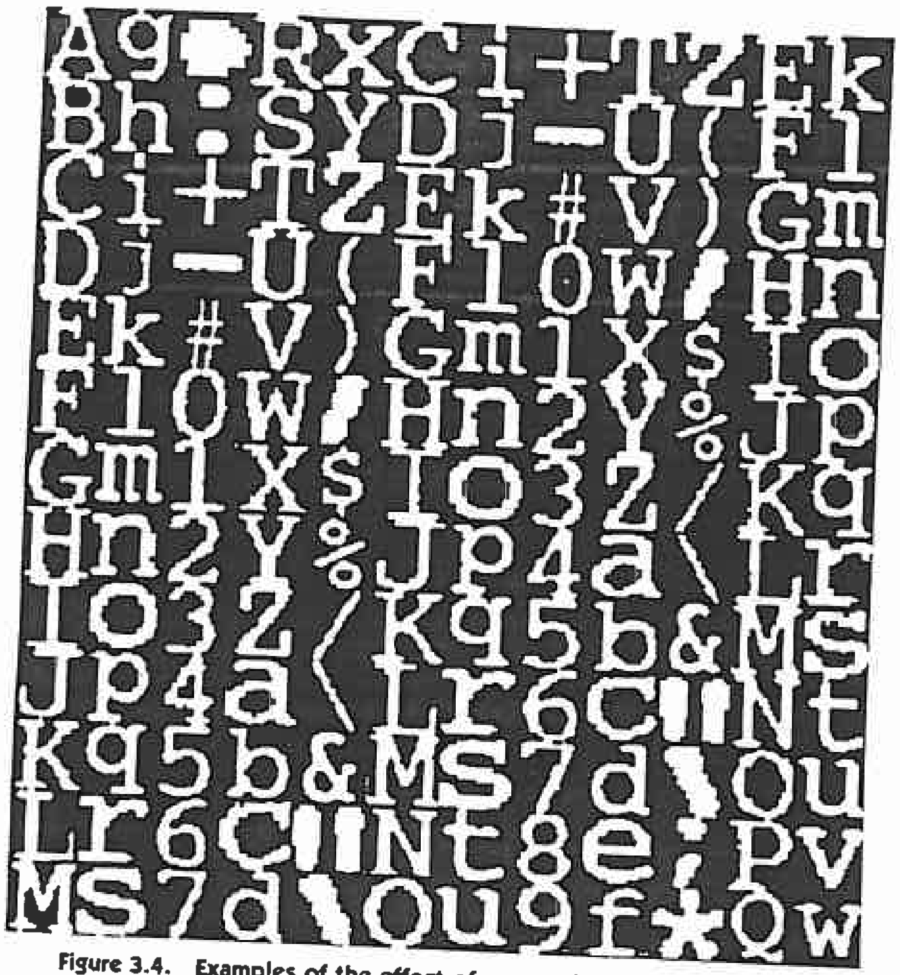

Figure 3.4. Examples of the effect of a normalization on characters.

the distribution of pixels near the top and bottom of the image. The centers of these two distributions are calculated and a line between the centers of them is calculated. This line is used to construct the edges of the imaginary parallelogram used in the shear. When a parallelogram around the character has been constructed, the rows of the image are shifted in the direction which will bring the edges of the parallelogram into vertical alignment. Typical results for four hand-printed digits are shown in Figure 3.5. The effect of this transform for machine print is usually negligible. Typical shear transform time is 90.1 μs per character.

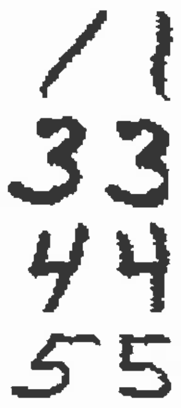
Figure 3.5. Examples of the effect of a shear transform on four characters.

2.3. Walsh Functions

Walsh functions in two dimensions are an orthogonal set of functions with amplitudes of ± 1; they can be used as the basis for a series expansion of a image in a way quite similar to the sine and cosine functions in a Fourier transforms. For the Walsh-function case the orthogonality condition is:

$$\mathbf{W}_{i,j} \cdot \mathbf{W}_{k,l} = \delta_{i,j,k,k'} \tag{1}$$

where $\delta_{i,j,k,l} = 0$, unless $i = k$ and $j = l$, when it equals one. This property allows an image, **q**, to be expanded in the form:

$$\mathbf{q} = \sum_{i}^{n_w} \sum_{j}^{n_w} A_{i,j} \mathbf{W}_{i,j}, \tag{2}$$

where

$$A_{i,j} = \mathbf{q} \cdot \mathbf{W}_{i,j}. \tag{3}$$

The details of this process are covered in [11]. Images of the Walsh functions to order 16 are shown in Figure 3.6. The black square in the upper left is the image of the function $\mathbf{W}_{1,1}$. Each square of this size is a function of increasing order in both the x and y directions. $\mathbf{W}_{5,1}$ the fifth-image square in the top row.

For a given image of size 32 × 32, the full set of 1,024 Walsh functions will reconstruct the image exactly. When only Walsh functions of order less than some order n_w are used a low-pass Walsh filter of order n_w results. When n_w is a

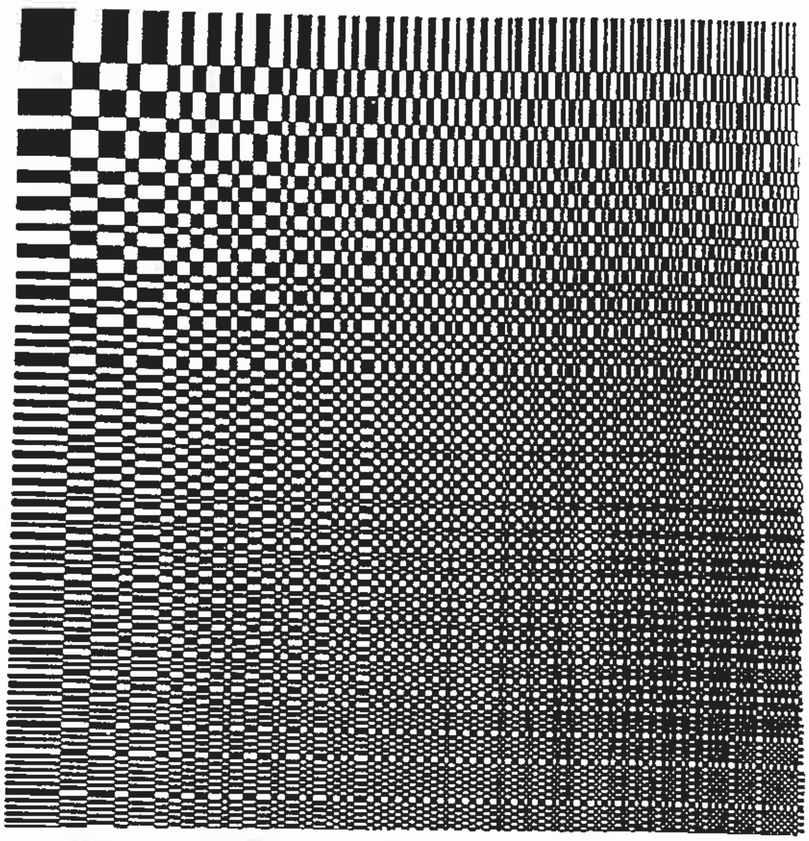

Figure 3.6. The Walsh basis functions in two dimensions out to order 16.

power of two, these low-pass filters result in an averaged down sampling of the image. If $n_w = 8$ for a 32×32 image, four-to-one down sampling results. The effect of low-pass filters of order 16, 13, 10, and 8 are shown in Figure 3.7. A typical Walsh filtering operation takes 6.1 ms.

2.4. Gabor Functions

The Gabor filtering section is accomplished using a least squares fit of each image. The kernel functions used are Gabor functions. The least squares fitting of the filter coefficients is necessitated by the nonorthogonal nature of the Gabor functions. Adopting the convention that bold upper case variables represent array-processor matrix data types and bold lower case variables represent array-processor vector data types, the Gabor functions are defined as:

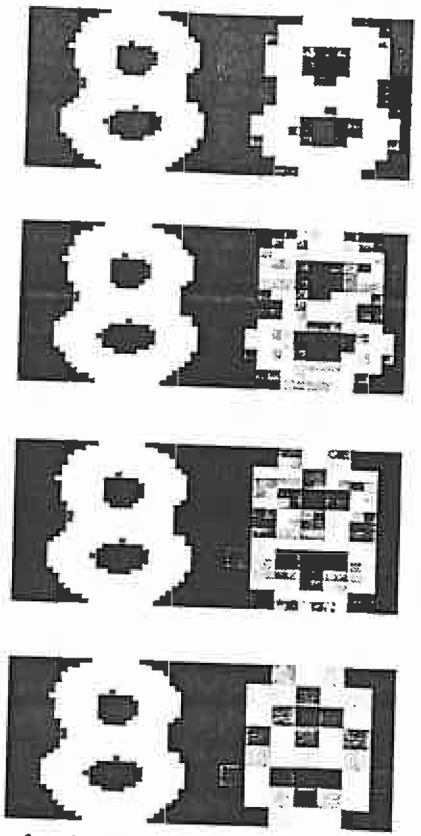

Figure 3.7. Examples of order 16, 13, 10, and 8 low-pass Walsh filters on the digit 8.

$$G_j(X, Y) = \exp(-R^2) \begin{cases} \sin(\omega_j X') \\ \cos(\omega_j X') \end{cases} \tag{4}$$

where the matrix variables **R**, **X′**, and **Y′** are given by:

$$R^2 = (X'^2 + Y'^2)/\sigma_j^2 \tag{5}$$

$$\begin{pmatrix} X' \\ X' \end{pmatrix} = T \begin{pmatrix} X - x_{0_j} \\ Y - y_{0_j} \end{pmatrix}. \tag{6}$$

The **X** and **Y** matrices in the array processor are row and column expanded in the form:

$$\mathbf{X} = \begin{pmatrix} x_1 & x_2 & \cdots x_{32} \\ x_1 & x_2 & \cdots x_{32} \\ & & \cdots \\ x_1 & x_2 & \cdots x_{32} \end{pmatrix} \tag{7}$$

$$\mathbf{Y} = \begin{pmatrix} y_1 & y_1 & \cdots y_1 \\ y_2 & y_2 & \cdots y_2 \\ & & \cdots \\ y_{32} & y_{32} & \cdots y_{32} \end{pmatrix} \tag{8}$$

A typical scalar transformation to be applied to each element of the matrix variables is a rotation of the form:

$$T = \begin{pmatrix} \cos\theta_j & \sin\theta_j \\ -\sin\theta_j & \cos\theta_j \end{pmatrix} . \tag{9}$$

The matrix function G_j is then expressed as a function of the scalar variables: ω_j, which is the spatial frequency of the function; σ_j, the spatial extent of the function; (x_{0_j}, y_{0_j}), the origin of the function; and θ_j, the orientation of the function.

2.4.1. Tiling

Since the Gabor basis functions are an infinite set, it is necessary to select a specific subset of them to be used as the filter elements which cover the character image. This selection process is referred to as tiling the image. For the class of filter discussed here each set of image origins has twice the sample density of the previous level and the number of directions selected, n_θ, is fixed. This results in a filter with directional sensitivity and positional sensitivity determined by the choice of the level parameter, i. The character images used in this study are 32×32, so that using large values of i would result in massive over-sampling of the image. The Gabor filter for the lowest value of i, on the other hand, is approximately a directional bar detector and adds little to the filter's spatial resolution. At each level the frequency and spatial resolutions, ω_i and σ_i, are adjusted to allow small overlaps in extent and provide octave spatial-frequency response.

After extensive experimentation, it was found that a reasonably good approximation to the image could be obtained by using only Level 2 Gabor functions. Reasonable directional selectivity was obtained with four-fold symmetry, $n_\theta = 4$. When the even and odd frequency components are included this results in a Gabor function set with 32 functions. All filter operations described in this chapter are carried out with these 32 function filters.

The results of the experiments are easily explained. For $i = 1$, the Gabor functions form a directional filter and provide only field-centered spatial loca-

Table 3.2. Table of the Possible Gabor Functions Used to Tile the Image. Each level, i, contains $2i^2n_\theta$ possible Gabor functions. The values of d_i are obtained by dividing the image as shown in Figure 3.8; $d_{i+1} = d_i/2$

i	x_{oi}	y_{oi}	ω_i	σ_i	θ_i
1	$\frac{d_0}{2}$	$\frac{d_0}{2}$	$\frac{2\pi}{d_0}$	d_0	$\frac{2\pi}{n_\theta}, \frac{4\pi}{n_\theta}, \ldots, 2\pi$
2	$\frac{d_1}{2}, \frac{3d_1}{2}$	$\frac{d_1}{2}, \frac{3d_1}{2}$	$\frac{2\pi}{d_1}$	d_1	$\frac{2\pi}{n_\theta}, \frac{4\pi}{n_\theta}, \ldots, 2\pi$
n	$\frac{d_n}{2}, \frac{3d_n}{2}, \ldots, \frac{(2^n - 1)d_n}{2}$	$\frac{d_n}{2}, \frac{3d_n}{2}, \ldots, \frac{(2^n - 1)d_n}{2}$	$\frac{2\pi}{d_n}$	d_n	$\frac{2\pi}{n_\theta}, \frac{4\pi}{n_\theta}, \ldots, 2\pi$

tion. As i increases, the resolution of the filter increases. At Level 3, the spatial and frequency resolution exceed the stroke size (line width) of the character and provide limited improvement in resolution. All experiments also suggest that, given the complex structure of Equations (4–9), sampling an image containing less than 16 pixels for each d_i interval is not an efficient use of Gabor functions.

2.4.2. Filtering

Once the Gabor functions are selected, the filtering operation starts by converting the binary image to an eight-bit image with a step height between levels of -127 and 127 with $\Sigma q = 0$. Since the set of Gabor functions is nonorthogonal, the filtering must be performed by least squares optimization. On the small images discussed here, direct methods are far more efficient for this operation than the neural net method proposed for data compression [4]. Given n different G_js the

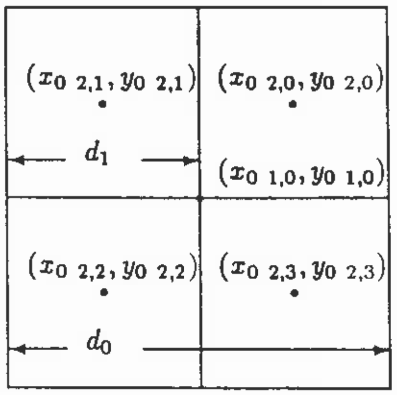
$(x_{0\ 2,1}, y_{0\ 2,1})$
$(x_{0\ 2,0}, y_{0\ 2,0})$
d_1
$(x_{0\ 1,0}, y_{0\ 1,0})$
$(x_{0\ 2,2}, y_{0\ 2,2})$
$(x_{0\ 2,3}, y_{0\ 2,3})$
d_0

Figure 3.8. Location of the first two levels of tiling points for the Gabor functions. The full set of locations is given in Table 3.2; in general $d_{i+1} = d_i/2$.

filtering operation is based on obtaining a least-squares fit to the image q by forming the Matrix A, each component of which is the inner product of the form:

$$a_{ij} = G_i \cdot G_j \tag{10}$$

and the vector,

$$b_i = q \cdot G_i \tag{11}$$

and solving

$$b = Ac \tag{12}$$

for the filter coefficients, c. Since the Matrix A is the same for any given set of n Gabor functions, the matrix is factored once, and only generation of b and back substitution of the factored A matrix is required to obtain each c. The image is converted to its filtered form:

$$q' = \sum_{j=1}^{n} c_j G_j \tag{13}$$

and then thresholded at zero making the image binary again.

The effect of the filter can be seen in Figure 3.9. The input image is converted to the gray-level image shown in the upper left quadrant of Figure 3.9. This input image contains 1,024 eight-bit elements. Using Equations (10–13), the reconstructed image q', shown in the upper right quadrant is produced. This image is constructed using 32 eight-bit values of c_j. The image is then thresholded at zero to yield the filtered image shown in the lower right quadrant of Figure 3.9. The residual error in the image fit, $q - q'$, is shown in the lower left quadrant of Figure 3.9. It is interesting to note that the residual output from the Gabor filter is similar in form to the output of an edge detector. This suggests that the Gabor filter is a "body-detecting" filter. The importance of this filtering will be discussed further when the impact of the filter on character recognition is considered.

An additional benefit of the feature enhancement properties of the Gabor filter can be seen in Figure 3.10. The input to the filter is the same as that shown in Figure 3.9, but with large quantities of random noise added, as shown in the upper left quadrant of Figure 3.10. The filtered image is shown in the upper right quadrant of Figure 3.10. The same thresholding procedure is used to produce the output shown in the lower right quadrant of Figure 3.10. The image in the lower left quadrant contains the residual and is a good image of the random noise in the image. The filtering was still done with 32 eight-bit adaptive coefficients.

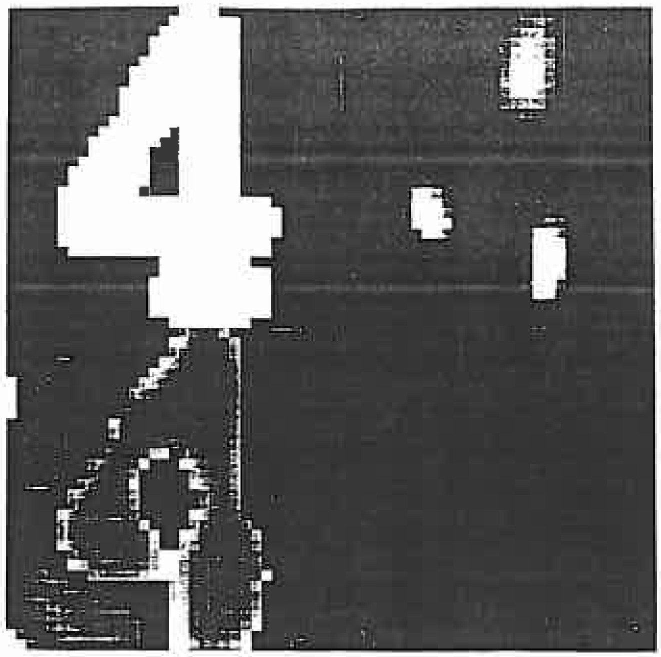

Figure 3.9. Gabor filtering of a character image.

3. ASSOCIATIVE RECALL

The data stored in the associative memories, M, is compared to the image, q. The association between a memory and an image is 0.0 for no similarity and 1.0 for a perfect match. The associative memories are initialized for the pattern with the memory value $M = 1$ and are initialized for the relevance with the value $M = 0$. Everything is true, but nothing is relevant. The maximum range on the eight-bit memories is $\pm S$.

The associative functions tested in the present implementation of FAUST are shown in Table 3.3. Five functions are used: (1) correlation, $1/(1 + \tan^2\theta$, $1/(1 + d^2)$, (2) Tanimoto similarity, (3) offset cosine, (4) cosine, and (5) Euclidian distance, d. Three of these functions (cosine, Euclidian distance, and Tantimoto) are discussed in [12]. The correlation-based method is the one used in [4]. The tangent-based function has a mathematical form similar to Tanimoto with the same properties near a perfect match as the distance-based form. All of the indicated sums are carried out over all pixels in the image.

The class of usable similarity functions is very large. Any monotonic, single-valued relationship between the image, q, and the memory contents, M, which can be conveniently mapped on to the interval zero to one, could be used.

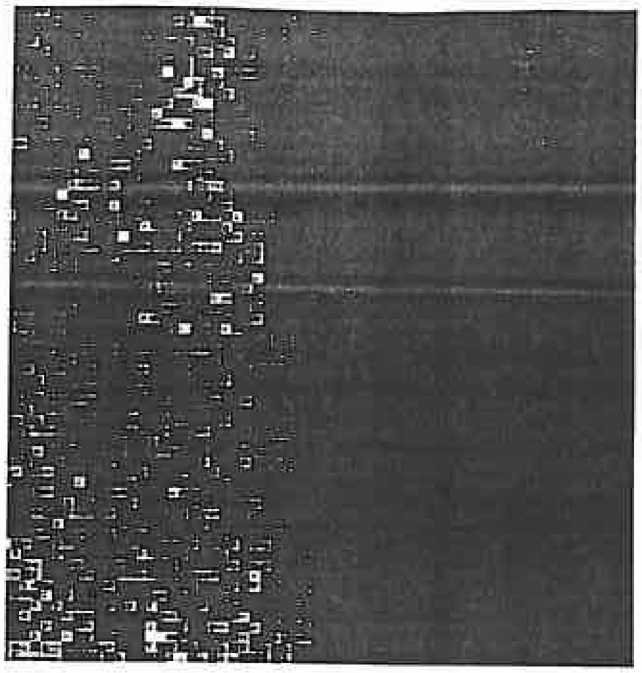

Figure 3.10. Gabor filtering of a character image with severe noise.

Table 3.3. Function Used for Associative Comparison of Binary Input Image, *q*, and memory, *M*, for Eight-bit and Binary Memory Data. Each image has N elements and eight-bit elements have a maximum value S. Binary input are treated arithmetically as being zero or one. Therefore $(-1)^q$ is 1 when $q = 0$ and -1 when $q = 1$.

Function	**Eight-bit Form**	**Binary Form**
Correlation	$-\Sigma(-1)^q M$	$\Sigma(q \equiv M)$
$1/(1 + \tan^2 \theta)$	$\dfrac{1}{1 + \dfrac{M^2 q^2 + (s\Sigma(-1)^q M)^2}{(-\Sigma(-1)^q M))^2}}$	$\dfrac{1}{1 + \dfrac{M^2 q^2 + (\Sigma(q \equiv M))^2}{(\Sigma(q \equiv M))^2}}$
$1/(1 + d^2)$	$\dfrac{1}{1 + \dfrac{\Sigma(M + (-1)^q s)^2}{4NS^2}}$	$\dfrac{1}{1 + \Sigma \sim (q \equiv M)/N}$
Tanimoto	$\dfrac{S\Sigma(-1)^q M}{M^2 + q^2 + S\Sigma(-1)^q M}$	$\dfrac{\Sigma(q \equiv M)}{2N - \Sigma(q \equiv M)}$
Offset Cosine	$\dfrac{\Sigma(-1)^q M + N}{2N\sqrt{\Sigma M^2} + \Sigma S^2}$	$(\Sigma(q \equiv M) + N)/(2\sqrt{2}N)$

The efficiency of any particular function will depend on the ability to separate correct classifications from weak associations on marginal images. In a character recognition application, the efficiency is partially counterbalanced by computational cost.

4. SYMMETRIC TRIGGERING OF LEARNING

During the learning process, images are presented, filtered. and compared with stored patterns using the association-strength equations of Section 3. If the match is adequate, the image is used to update the memory in the learning phase. If the image is not sufficiently similar to the existing patterns then it is treated as a unique pattern and placed in a new memory location. Symmetric triggering is the concurrent logical operation which uses comparable logical structures to update all of the associative memory blocks in parallel. This parallel operation occurs in an association space which has a dimension equal to the number of feature classes and associative memory blocks.

The most general form of the FAUST-triggering logic for N associative memories for each feature class and M feature classes, for a total of $N \times M$ memories which each have association strengths, $A_{i,j}$, and logical thresholds for triggering, ρ_j, where $j = 1, \ldots, M$ and $i = 1, \ldots, N$ is

$$\max_i \left(\sum_{j=1}^{M} (A_{i,j} - \rho_j)^2 \right) \;\&\; (A_{i,1} > \rho_1) \cdots \;\&\; (A_{i,j} > \rho_j) \cdots \;\&\; (A_{i,N} > \rho_N) \,. \quad (14)$$

The learning is triggered symmetrically in the ith set on N memories, and N features are being learned across M feature classes.

This is shown in two dimensions, $M = 2$, in Figure 3.11.

A less general type of triggering, similar to the ART methods, is obtained if X takes on values R and P and $N = 2$. Pattern existence data is represented by n binary map-feature classes indexed on j and with i occurrences of each pattern, $P_{i,j}$, with association strength, $P_{i,j}$. Triggering is initiated by n vigilance parameters, ρ_j. These vigilance parameters are not counted in the left-hand distance term and pattern strength. Relevance data is represented by m multibit map-feature classes indexed on k with i occurrences of each strength class, $R_{i,k}$, with association strength, $R_{i,k}$, which have $\rho_k = 0$. The generalized FAUST logic for triggering learning takes the form: Find an i and $\mathrm{P} = \Pi_{j=1}^{n}(P_{i,j}) \times \Pi_{k=1}^{m}(R_{i,k})$ using

$$\max \left(\sum_{k=1}^{m} (R^2_{i,k}) \right) \;\&\; (P_{i,1} > \rho_1) \cdots \;\&\; (P_{i,j} > \rho_j) \cdots \;\&\; (P_{i,n} > \rho_n) \,. \quad (15)$$

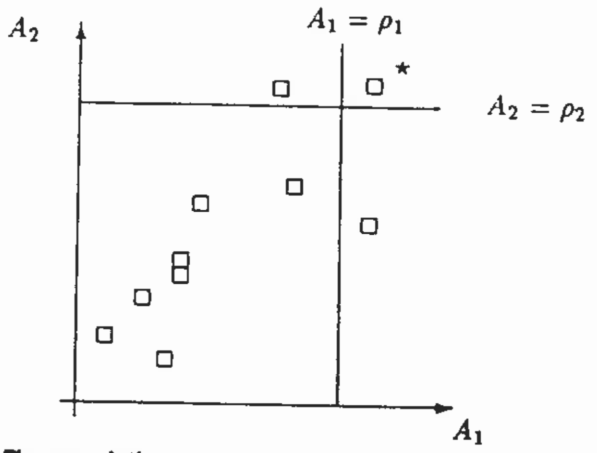

Figure 3.11. **The association space diagram that is used for symmetric triggering. The □s are for nonmatching points. The ★ is for the correct classification. The limits $A_{i,j} > \rho_j$ for $j = 1, 2$ are also marked.**

For the case discussed in the example presented here, $j = 1$ and $k = 1$, so that the FAUST-triggering logic reduces to: find an i using

$$\max(R_i^2) \; \& \; (P_i > \rho). \tag{16}$$

A typical association space diagram for a set of R_i and P_i points are shown in Figure 3.12 for $i = 1, 2, \ldots, 10$. The area P for the maximum case is marked

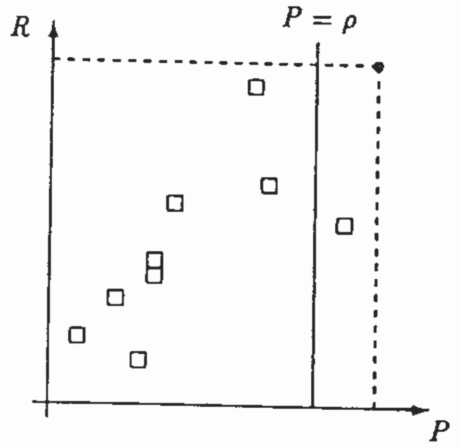

Figure 3.12. **The association space diagram that is used for symmetric triggering. The □s are for nonmatching points. The • is for the correct classification. The area P and the limit $P_i > \rho$ are also marked.**

by the dashed lines. The limit $P_i > \rho$ is marked by a vertical line. The case which triggers learning is marked by a bullet, •. Weaker associations which do not result in learning are shown as boxes, □.

5. LEARNING METHODS

After learning is triggered, information from the image, q, is stored in one of the eight-bit memories used by the relevance feature class, relevance memory, and into one of the binary memories used by the pattern feature class. The location used is determined by the logic discussed in Section 4. Any absolutely stable and convergent learning rule may be used. Six different learning rules of varying complexity have been used for the relevance memories; two different rules have been used for the pattern memories. During learning, the class of the sample images is not used. The process is self-organizing and requires no knowledge of class to construct the learned images. No explicit knowledge of class is used except in the statistical evaluation of classes discussed in Section 6.3.

5.1. Learning in Multibit Memories

All of the learning methods used for eight-bit relevance memory take the form:

$$M_{i,j}(t + 1) = g(M_{i,j}(t) + \alpha\Delta M_{i,j}), \tag{17}$$

where if $\pm S$ is the scale, maximum range, of M, the limit function g is given by:

$$g(x) = \begin{cases} S & \text{if } x > S \\ x & \text{if } S \geq x \geq -S, \\ -S & \text{if } x < -S \end{cases} \tag{18}$$

α is the learning rate, and the memories are updated form epoch t to epoch $t + 1$. The time dependence generates a dynamically stable learning sequence. All of the rules used here involve mechanism which generate positive feedback between the memory locations and the image. The function g is used to provide a bound on this feedback.

The first rule is the Dystal [10] rule given by:

$$\Delta M_{i,j} = \begin{cases} S - M_{i,j}(t) & \text{if } q_{i,j}(t + 1) > 0 \\ M_{i,j}(t) - S & \text{otherwise.} \end{cases} \tag{19}$$

This rule was developed in [10] to more closely approximate the behavior of neurons than previous rules. This has important consequences for the stability of the learning process. Unlike the other rules used here, this rule is self-limiting at the scale values $\pm S$. This rule is only applied vertically; data from each pixel only effect one memory element during learning. When a more extended field is used, image data effects memory elements over some local region, referred to as a receptor field. A rule of the Dystal type has a distinct stability advantage in the receptor field cases over nonlinear limit functions.

The second rule used is a simple Hebbian rule [13] of the form

$$\Delta M_{i,j}(t) = \begin{cases} S & \text{if } q_{i,j}(t+1) > 0 \\ -S & \text{otherwise.} \end{cases} \tag{20}$$

This rule is used because it is simpler and faster than the rules used in [7] and [8] but can be comparisons with these rules. The stability on the rule is guaranteed since only vertical interconnections are used.

The third rule used was a vertically distributed Hebbian rule [7] of the form

$$\Delta M_{i,j}(t) = \sum_{k=i-1}^{k=i+1} \sum_{l=j-1}^{l=j+1} \begin{cases} S & \text{if } q_{k,l}(t+1) > 0 \\ -S & \text{otherwise.} \end{cases} \tag{21}$$

The vertical part of the learning is identical to Rule 2. The field used is a 3×3 square. The parallel processor implementation is shown in Figure 3.13. The arrows indicate shifting operation on the processor array. Although a loop structure is implied by the notation used for the rule, a linear sequence of shifts is used. The stability of this rule is discussed in the next section.

The fourth rule used was a vertically distributed Hebbian rule and a laterally connected anti-Hebbian rule [8] of the form

$$\Delta M_{i,j} = \begin{array}{l} \sum_{k=i-1}^{k=i+1} \sum_{l=j-1}^{l=j+1} \begin{cases} S & \text{if } q_{k,l}(t+1) > 0 \\ -S & \text{otherwise.} \end{cases} \\ -\sum_{k=i-1}^{k=i+1} \sum_{l=j-1}^{l=j+1} \begin{cases} S & \text{if } \operatorname{sgn}(M_{k,l}(t)) \equiv \operatorname{sgn}(M_{i,j}(t)) \\ -S & \text{otherwise,} \end{cases} \end{array} \tag{22}$$

where

$$\operatorname{sgn}(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{if } x = 0 \\ -1 & \text{if } x < 0. \end{cases} \tag{23}$$

The vertical part of the learning is identical to Rule 3. The lateral component is anti-Hebbian. Both the vertical and lateral terms used the field-distribution pattern, shown in Figure 3.13.

The fifth rule used was a vertically-distributed Hebbian rule with Gaussian

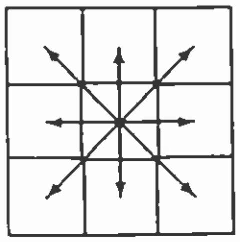

Figure 3.13. Distribution pattern used in the 3 × 3 field profiles of rules 3 and 4.

weights [7] on a field n_f square which must be summed over $q = (n_f - 1)/2$ elements and takes the form

$$\Delta M_{i,j}(t) = \sum_{k=i-q}^{k=i+q} \sum_{l=j-q}^{l=j+q} \begin{cases} \exp(-(i^2 + j^2)/\sigma^2)S & \text{if } q_{k,l}(t+1) > 0 \\ -\exp(-(i^2 + j^2)/\sigma^2)S & \text{otherwise,} \end{cases} \tag{24}$$

where σ is the envelope variance of the field; this variance is not related to the variance used in the Gabor filter in Equation (5). The field-distribution pattern used for a version of this rule in which $n_f = 7$ is shown in Figure 3.14. The addition of the Gaussian weights increases the stability of the learning by making the equivalent shunting matrix, discussed in Section 6, diagonally dominant.

The sixth rule used was a vertically distributed Hebbian rule and a laterally connected anti-Hebbian rule with Gaussian weights [8] on field n_f square which must be summed over $q = (n_f - 1)/2$ elements and takes the form

$$\Delta M_{i,j} = \begin{array}{l} \sum_{k=i-q}^{k=i+q} \sum_{l=j-q}^{l=j+q} \begin{cases} \exp(-(i^2 + j^2)/\sigma^2)S & \text{if } q_{k,l}(t+1) > 0 \\ -\exp(-(i^2 + j^2)/\sigma^2)S & \text{otherwise} \end{cases} \\ -\sum_{k=i-q}^{k=i+q} \sum_{l=j-q}^{l=j+q} \begin{cases} \exp(-(i^2 + j^2)/\sigma^2)S & \text{if } \operatorname{sgn}(M_{k,l}(t)) \equiv \operatorname{sgn}(M_{i,j}(t)) \\ -\exp(-(i^2 + j^2)/\sigma^2)S & \text{otherwise,} \end{cases} \end{array} \tag{25}$$

where σ is again the envelope variance of the field. The field-distribution pattern used for this rule is shown in Figure 3.14.

5.2. Learning in Binary Memories

The two rules used for binary pattern learning are a logical "OR" of the positively relevant elements

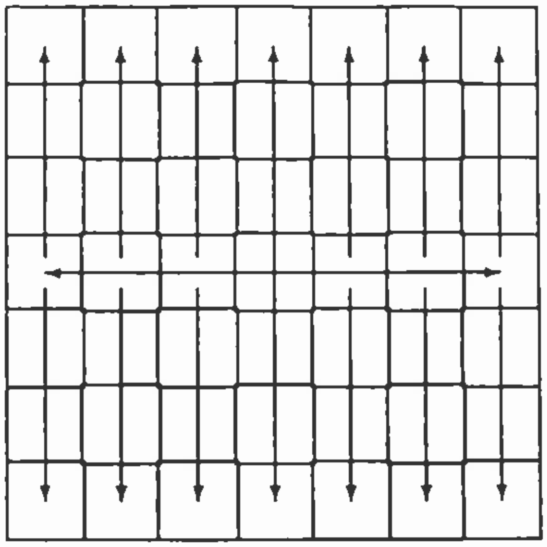

Figure 3.14. Distribution pattern used in the 7 × 7 field profiles of rules 5 and 6.

$$M_{i,j}(t+1) = \begin{cases} M_{i,j}(t) \vee q_{i,j}(t+1) & \text{if relevance} > 0 \\ M_{i,j}(t) & \text{otherwise,} \end{cases} \tag{26}$$

and a logical product of the image and relevant elements.

$$M_{i,j}(t+1) = \begin{cases} 1 & \text{if } q_{i,j}(t+1) > 0 \text{ and relevance} > 0 \\ 0 & \text{if } q_{i,j}(t+1) = 0 \text{ and relevance} < 0 \\ M_{i,j}(t) & \text{otherwise.} \end{cases} \tag{27}$$

6. PATTERN CLASSIFICATION

Two types of classification have been used with FAUST to provide classification of images based on a previously learned FAUST image-feature set. In the simplest case, maximum resonance-based classification was used as in [3] with ART-1 features. Since the association space diagram generated by the FAUST feature set is more selective than the associations, which are generated by ART-1, a more general type of classification based on analysis of the association space diagram is also discussed.

6.1. Resonance-Based Classification

If M_R is a relevance memory and M_P is a pattern memory. the classification of an image, I, is found by finding

$$\max(-\Sigma((-1)^I M_R \,\&\, (I \equiv M_P)) \tag{28}$$

and then taking the classification of the majority of the images used to learn the $\{M_R, M_P\}$ pair. Since each of the terms used to generate the sum is equivalent to the summed term in the correlation-association rule shown in Table 3.3, the function maximized is the cross correlation of the two memories.

In the limiting case where the edge pixels have relevance zero then this would be represented by Figure 3.15. When the relationship of the two memories is contradictory near the edge, the area would over-represent the cross-correlated relevance. In the case of complex learning rules, such as Rule 4 in Section 5, the area would also be decreased by the down weighting of uniform regions far from any edge.

6.2. Association-Based Classification

The association-based method calculated the values of R and P according to one of the functions used in Section 3. These values are then used to compute mean values $\bar{R}$ and $\bar{P}$. These values are then used to find:

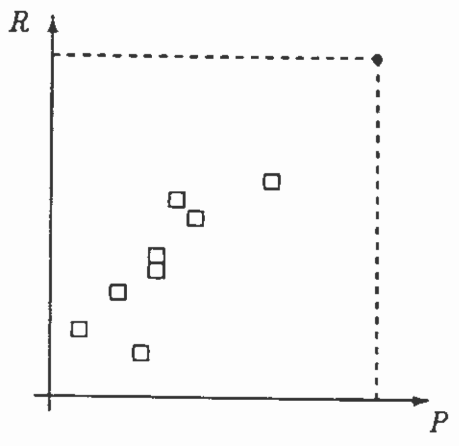

Figure 3.15. The association space diagram that is used for maximum-resonance classification. The □s are for nonmatching points. The • is for the correct classification.

$$\max\left(\sqrt{(R - \bar{R})^2 + (P - \bar{P})^2}\right) \qquad (29)$$

for which $R > \bar{R}$ and $P > \bar{P}$. This is illustrated by the vector between the mean and classifying point in Figure 3.16. As in the previous case, the classification is determined by the majority of the images used to learn the classifying memory pair.

6.3. Statistical Evaluation of Classes

The evaluation of the self-organized classes is achieved by accumulation of statistics in a classification variable:

$$Z_{class,k,j} = Z_{class,k,j} + 1 \text{ if class of } (q_i) = k. \qquad (30)$$

This table can then be used to determine the most likely classification of each associative memory of the set for all images and to assign classes to images based on maximum strength achieved using either Equations (28) or (29) over all memory location and class assignments. This allows a new set of images to be learned wholly by example and divided into classes based on the recognition results achieved using the images assigned to the training set. This procedure is equivalent to applying an inhibiting signal to the gates in the diagram in Figure 3.2.

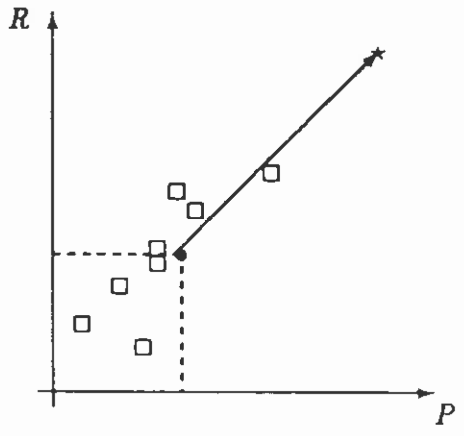

Figure 3.16. The association space diagram that is used for associative classification. The □s are for nonmatching points. The ★ is for the correct classification. The • marks the mean.

7. STABILITY IN FINITE PRECISION IMPLEMENTATIONS

All of the learning rules discussed in Section 5 are of the approximate form:

$$M(t + 1) = M(t) + \Delta M. \tag{31}$$

This can be reformulated, to match as far as possible the notation of [14], in the form:

$$\Delta M = (B - CM)Q - (DM + E)J \tag{32}$$

where the shunting network, that is the distributive mechanism which generates a feedback equivalent to the bounding function, for excitatory inputs, where $Q = u^T q$, and $u = (1, 1, \ldots, 1)$ requires that signals be ineffective when

$$B = CM \tag{33}$$

and in the present model where $Q = J$ the shunting of inhibitory input occurs when

$$E = DM. \tag{34}$$

In [14] it is shown that absolute stability results when the matrices C and D are symmetric. This reformulation, as discussed in [14], is equivalent to replacing the limiting function $g(x)$ by feedback terms of the form CMQ and DMJ where C and D are chosen to limit ΔM to values which constrain $M(t + 1)$ to the range $\pm S$.

In the present work, M, q, B, E, and j are 2^s, s-bit, vectors of length 1,024, and C and D are $1{,}024 \times 1{,}024$ square 2^s, s-bit, symmetric matrices. If the restriction on the significance of these elements to s-bits did not apply then the proof given in [14] would be sufficient to guarantee absolute stability.

A simple construction will show the consequences of finite precision on the stability criteria. Take $s = 8$, $\alpha = 1.0$, and assume for Rule 5 of Section 6 that $\sigma^2 >> i^2 + j^2$ and $n_f = 7$. Under these conditions the Gaussian coefficient approaches 1.0 and each double sum has 49 terms, each with eight bits. The accumulation of one row of B times Q requires $2^6 \times 2^8$ or 2^{14}, 14-bit, arithmetic. This calculation will work with 16-bit accumulators. In general, the accumulation of one row will require:

$$n_{acc} = s + 2 \log_2(n f) + \log_2(\alpha) + 1 \tag{35}$$

bits. The learning rate, α, is less than or equal to one so that lower learning rates can decrease the need for large accumulators by allowing stepwise averaging.

The solution to the shunting equation poses more difficult numerical problems. If the calculation is done in fixed point, then the worst-case condition for formation of a term of BC^{-1} will be when $c_{i,j} = 1/b_{i,j}$ which will require $2n_{acc}$ bit dynamic range. For floating point calculation, the mantissa must contain $2n_{acc}$ bits to avoid round-off errors. This would require 28-bit arithmetic, nine decimal digits, to avoid round off for a seven-element receptor field. A fully connected field would require 38 bits of precision. Decreasing the learning rate will decrease the number of required bits by 2 $\log_2(\alpha)$. For $\alpha = 1/8$, the 7×7 field would then require 32 bits.

The formation of the row accumulations and shunting terms are not the most likely cause of errors in the numerical calculation. The most difficult computational task is the solution of the two shunting equations and particularly the inhibitory equations resulting from Rules 4 and 6. The numerical theory required to analyze these solutions is described in [15] and is outside the scope of this work. The results are summarized by stating that for all rules the Matrix C is symmetric, positive definite, and diagonally dominant. The Matrix D from Rules 4 and 6, for regions which are already inhibited, may result in poorly conditioned values of D.

8. RESULTS

The methods described in Sections 2–6 were tested by constructing a recognition system which can adaptively learn both machine and hand-printed digits on a massively parallel processor. The parallel FORTRAN program is approximately 1,400 lines long. The hardware used is discussed in Section 8.1 (below). A test sequence consists of loading the images of the test data into the array processor and performing filtering, feature extraction, and classification on these images. Each test set is divided in half. The first half is used to learn the character set features using the FAUST associative memories. The second half is used for classification testing of the degree of generalization achieved. All classifications use the method described in Section 6. All error rates shown are for the second half of the test sample.

The symbols used in all tables are:

1. *In* is the sample file set name,
2. *M* is the number of memories,
3. *N*—normalize, *S*—Shear, and *G*—32 term Gabor (sequentially applied filter types),
4. ρ is the vigilance,
5. *P* is the pattern association type (*C*—correlation, *D*—inverse squared distance, *Tan*—inverse squared tangent, *T*—Tanimoto, *Cos*—offset cosine),
6. *R* is the relevance association (types as for *P*)

7. *PI* is the pattern learning rule (*I*—logical or. *R*—relevance controlled),
8. *RI* is the relevance learning rule (*D*—Dystal. *H*—vertical Hebbian, *V*—nine point vertical Hebbian, *Hz*—nine point vertical Hebian and nearest neighbor horizontal anti-Hebbian, *L5*—5 × 5 Gaussian enveloped vertical Hebbian. *H5*—5 × 5 Gaussian Hebbian and 5 × 5 Gaussian horizontal anti-Hebbian,
9. *Wro* is the number of substitutional errors.
10. *Unk* is the number unknown,
11. *Confidence* is the confidence for correct answers.
12. *Failure Con.* is the confidence for wrong answers. and (The test sample file have the size and composition shown in Table 3.4.)

The four image-processing requirements discussed in the introduction are achieved by:

1. scale invariance—normalization,
2. rotational invariance—shear transform,
3. edge position invariance—FAUST relevance learning, and
4. local shape invariance—Gabor filtering.

The need for Items 2 and 4 is limited to hand print. Machine fonts are sufficiently square on the page and uniform in shape so that only normalization and relevance learning are needed.

8.1. Hardware Architecture

The parallel data flow, shown in Figure 3.2, is ideally suited to a massively parallel architecture in which the number of processors is equal to or exceeds the number of input data elements. Since many of the operations on the input involve only limited precision operations, the precision of each processor need not be

Table 3.4 Table of the File of Data Used in Testing. Each hand-print sample comes from a different writer in [16].

name	type	size
fl1	machine print	300
fl2	hand print	300
fl3aa	hand print	1024
fl3ab	hand print	1024
fl3ac	hand print	1024

great. Also, since data flow between the various parts of the architecture is critical, the bandwidth for communications between processors must be high.

The SIMD (Single Instruction Multiple Data) architecture used for this study meets all of these requirements. The computer used was an Active Memory Technology 510 Distributed Array Processor [12], DAP510.[1] This machine consists of a 32 × 32 grid of one bit processor element (PE). Operation of the PE array is controlled by a 4 MIPS RIS master control unit (MCU). All program instructions are stored in a separate code memory and are passed to the PE array through the MCU. A block diagram of this architecture is shown in Figure 3.17.

All data is stored in a separate array memory. The array memory is organized in 32 × 32 bit planes with each of the bits in each plane connected to one XPEData can also be passed between PEs along the grid. The cycle time of all PEs is 100 ns.

This processor configuration is capable of performing 10^{10} binary operations per second; processing speed decreases proportionally with the length of data items used. Two data mappings are particularly well suited to the DAP structure: a vector mode in which successive bits of a single word are mapped into a row of the array, and a matrix mode in which successive bits of a word are mapped into layers of the array memory vertically. Both of these modes of operation are used in FAUST. The summation operations shown in Figure 3.2 use floating-point matrix-mode operations to trigger learning and logical matrix operations to learn binary pattern information.

8.2. Learning

The learning process in FAUST is illustrated in Figures 3.17, 3.18, and 3.19. In Figure 3.17 the pattern and relevance memories are shown in the two left columns and the characters used in the learning process are shown extending to the right of each memory pair. In Figure 3.18 the character images are replaced by plots which are miniature versions of the association space diagrams in Figures 3.11, 3.15, and 3.16. In these plots the point at the upper left is the correct match. Figure 3.19 is an enlarged view of material from the top row, where the "3" is learned. In the upper left the learned pattern is shown. In the upper right the learned relevance is shown. The gray pixels represent regions of the character where uncertainty has been detected and learned. In the lower left of Figure 3.19 the last character learned is shown. In the lower right, the association space map is shown.

[1] Certain commercial equipment may be identified in order to adequately specify or describe the subject matter of this work. In no case does such identification imply recommendation or endorsement by the National Institute of Standards and Technology, nor does it imply that the equipment identified is necessarily the best available for the purpose.

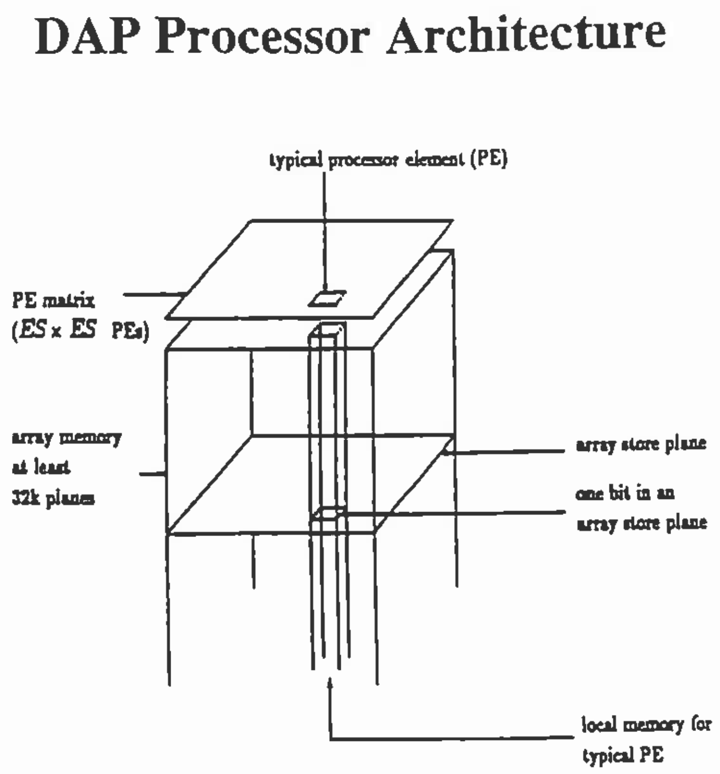

Figure 3.17. Array processor architecture for massively parallel computer.

The way FAUST learns is demonstrated in Figure 3.17 by the increase in image contrast in Column 2, the relevance memory data. Initially all locations contain a blank gray field. As learning progresses, regions of the image which are dark decrease in intensity and regions which are light increase in intensity. The "9" in the twelfth row has been exposed to only one learning event and contains a low contrast image. The "2" in the thirteenth row has been exposed to two learning events and shows increased contrast. The "7" in Row 6 has been exposed to enough learning to generate a high contrast image.

This shows gradual learning, but the importance of the learning process is shown in the association space diagrams in Figure 3.18. The x-axis on these diagrams is the pattern association strength. The y-axis is the relevance association strength. As learning progresses, classification accuracy is improved if one point in the diagram separates from the other clustered points. In Row 4 the "4" recognition is achieved by the point in the upper right of each diagram. Most of the separation, which results in improved classification, is achieved on the relevance axis.

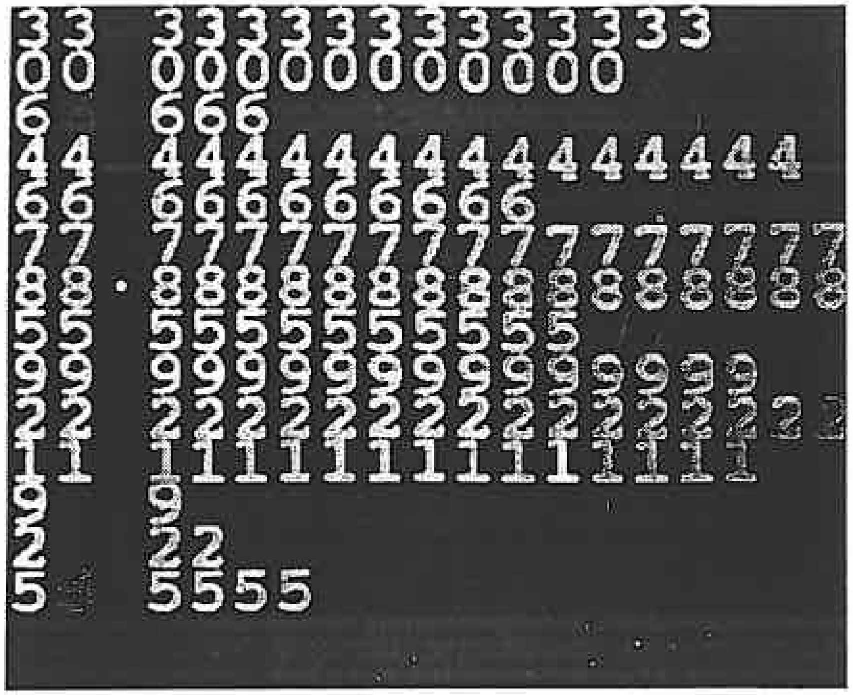
Figure 3.18. Plot of learned memory material, the learned pattern is in column 1, the learned relevance is in column 2, and the characters used in FAUST for the learning are in columns 3 and above.

The mechanism which achieves this separation is shown in the expanded views in Figure 3.19. The relevance memory data for the "3" in row one, shown in the upper-right quadrant, is dark in the background, bright inside the digit, and gray at the edges. The uncertainty which has been learned at the edges is responsible for the increased relevance association strength during learning. Digits are very reliable over most of the image. The difference in images is in the body of the character.

8.3. Machine-Print Data

8.3.1. Effect of association rule

Tables 3.5 and 3.6 show the effects of different association rules on machine-print character recognition. All five rules are used. In all cases, with the correct choice of ρ, it is possible to achieve 100% recognition on test samples of 150 characters. The association rules effect the sensitivity of learning and through

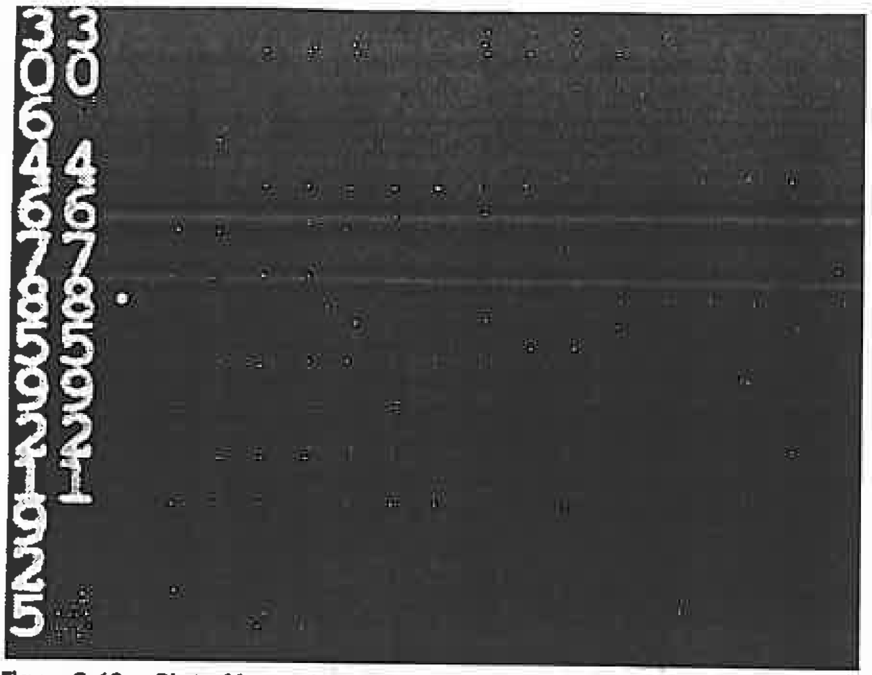

Figure 3.19. Plot of learned memory material, the learned pattern is in column 1, the learned relevance is in column 2, and the association space diagrams which result from FAUST for the learning are in columns 3 and above.

them the confidence levels. This is caused by the variation in sensitivity of the functions in Table 3.3 to image similarity differences. The minimum recognition rate is achieved using inverse square distance association and resonance classification and is 2.4 ms/character.

8.3.2. *Effect of classification method*

The differences in Table 3.5 and 3.6 are caused by using different classification rules. The same recognition accuracy is achieved in each case, but the confidence, both of correct and of failed recognitions, is higher for resonance classification. The advantage of associative classification is that the ratio of correct to failure confidence is larger. This allows better discrimination between correct and unknown cases. This improvement is achieved with slower recognition. The recognition rate ranges between 4.7 ms per image for correlative association to 12.6 ms per image for offset cosine association.

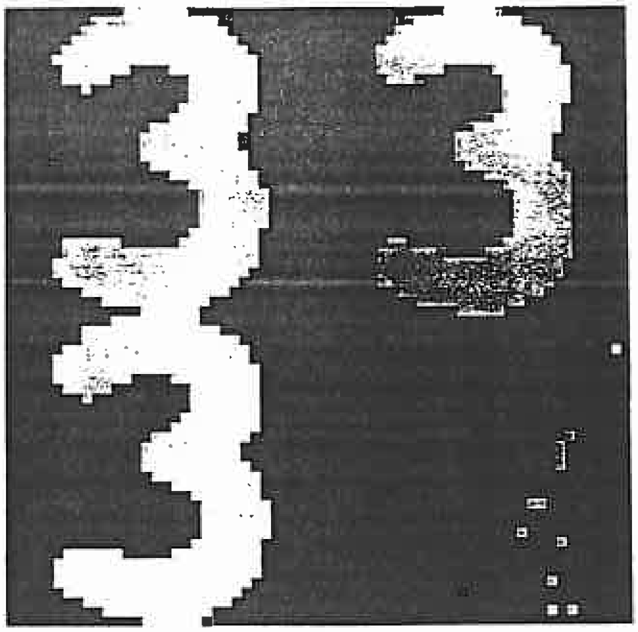

Figure 3.20. Enlarged plot of learned pattern memory and relevance memory material, last characters used, and last association plot used in FAUST learning.

8.4. Hand Print

8.4.1. Effect of filters

Table 3.7 shows the classification error for three different samples of 512 hand-printed digits. The digits were taken from hand-printed digits contained in the NIST hand-print database [16]. Data from 18 different individuals were used in

Table 3.5. Machine-Print Classification Results. Legend is as in Section 8. Resonance classification was used.

In	M	Filter	ρ	P	R	Pl	Rl	Wro	Unk	Confidence	Failure Con.
fl1	16	N	0.9	C	C	R	H	none	none	0.961225	no data
fl1	16	N	0.9	Tan	Tan	R	H	none	13	0.955444	0.736723
fl1	16	N	0.8	Tan	Tan	R	H	none	none	0.96159	no data
fl1	16	N	0.9	D	D	R	H	none	none	0.960049	no data
fl1	16	N	0.9	T	T	R	H	none	13	0.954134	0.736723
fl1	16	N	0.9	Cos	Cos	R	H	none	none	0.961225	no data
fl1	16	N	0.8	T	T	R	H	none	none	0.960049	no data

Table 3.6. Machine-Print Classification Results. Legend is as in Section 8. Associative classification was used.

In	M	Filter	ρ	P	R	PI	RI	Wro	Unk	Confidence	Failure Con.
fl1	16	N	0.9	C	C	R	H	none	none	0.642354	no data
fl1	16	N	0.9	Tan	Tan	R	H	none	13	0.562907	0.04142
fl1	16	N	0.8	Tan	Tan	R	H	none	none	0.60939	no data
fl1	16	N	0.9	D	D	R	H	none	none	0.597593	no data
fl1	16	N	0.8	T	T	R	H	none	none	0.58968	no data
fl1	16	N	0.9	Cos	Cos	R	H	none	none	0.782599	no data

each test file. Nine individuals were used in the learning phase, and a different nine individuals were used to test classification. Several different filter types were used on the hand-printed characters. Undetected error rates are a minimum of 4.9% and detected error rates are about 12.1%. The use of resonance classification is less effective than associative classification. The most effective filter combination is shown to be a shear transform followed by a Gabor filter. These combined filters reduce the substitutional error by a factor of three and the number of unknowns by a factor of two.

8.4.2. Effect of associative classification

As in the machine-print case, the effect of using associative classification is to increase the accuracy of the recognition process. Only the shear and Gabor filter types were used since they were the most effective with the resonance classification. Undetected error rates are a minimum of 2.3% and detected error rates are

Table 3.7. Classification Errors in a Sample of 512 Hand-Printed Characters When Several Different Filters Are Used in FAUST. Resonance classification and nearest neighbor Hebbian learning are used throughout.

Sample	Memory	Filter	ρ	Wrong	Unknown	Confidence	Failure Con.
fl3aa	64	None	.8	83	106	.837977	.810095
fl3aa	64	S	.8	66	83	.847505	.815503
fl3aa	64	G	.8	56	80	.814812	.765439
fl3aa	64	S + G	.8	25	62	.822200	.765033
fl3aa	64	S + G	.85	25	57	.830161	.767745
fl3aa	64	S + G	.90	44	66	.841545	.78179
fl3aa	64	S + G	.75	54	105	.828508	.776248
fl3ab	64	S + G	.85	30	91	.833265	.756408
fl3ac	64	S + G	.85	31	72	.832459	.776401
fl3aa	32	S + G	.85	39	88	.814784	.722403
fl3ab	32	S + G	.85	44	103	.814545	.742161
fl3ac	32	S + G	.85	63	104	.817895	.761885

Table 3.8. Associative Classifier Results. Legend is as in beginning of Section 8.

In	M	Filter	ρ	P	R	Pl	Rl	Wro	Unk	Confidence	Failure Con.
fl2	32	none	0.8	C	C	R	V	10	37	0.227571	0.149181
fl2	32	S + G	0.8	C	C	R	V	4	23	0.283133	0.155104
fl3aa	64	S + G	0.8	C	C	R	V	24	67	0.343192	0.202113
fl3aa	64	S + G	0.8	Tan	Tan	R	V	24	69	0.263423	0.128439
fl3aa	64	S + G	0.8	D	D	R	V	none	458	0.0001	0.0001
fl3aa	64	S + G	0.8	T	T	R	V	17	90	0.246038	0.111331
fl3aa	64	S + G	0.8	Cos	Cos	R	V	none	455	0.0	0.0
fl3aa	64	S + W16 + G	0.8	C	C	R	V	22	68	0.356265	0.20178
fl3aa	130	S + G	0.8	C	C	R	V	19	60	0.352834	0.220397
fl3aa	130	S + G	0.8	C	C	R	V	12	48	0.368763	0.217061
fl3aa	130	S + G	0.8	C	C	R	L5	27	56	0.361662	0.259817
fl3ab	130	S + G	0.8	C	C	R	V	27	51	0.377148	0.29399
fl3aa	130	S + G	0.85	C	C	R	V	14	62	0.391238	0.211646
fl3aa	130	S + G	0.75	C	C	R	V	33	67	0.340328	0.226505
fl3ab	130	S + G	0.8	Tan	Tan	R	V	16	70	0.290596	0.147452
fl3ab	130	S + G	0.8	T	T	R	V	15	68	0.29219	0.15415
fl3aa	130	S + G	0.8	T	T	R	V	12	76	0.301668	0.142273
fl2	130	S + G	0.8	T	T	R	V	4	14	0.271369	0.150463

about 9.4%. The substitutional error has been cut in half by the use of associative classification and the recognition rate has improved from 83 to 88%. Since the number of filter types, association types, learning rules, and classification rules form a large set, which has not been completely explored, better values for recognition rate may be possible.

9. CONCLUSIONS

A new architecture, FAUST, has been developed which provides a method for performing self-organizing pattern recognition using multimap classification and learning similar to the multimap structures known to exist in the vertebrate sensory cortex. The method is well suited to implementation on massively parallel computers, since learning the relevance of image regions is in parallel with the learning of image data itself. This has allowed a complete character-recognition package to be designed which uses unsupervised biologically motivated methods for input filtering, feature extraction, and classification.

The effectiveness of the FAUST character-recognition methods are compared to several other biologically motivated methods in Table 3.9. All of the calculations were performed on an Active Memory Technology DAP 510. The data for the first three methods is taken from [18]. All of the timing data is for recognition

Table 3.9. Comparison of the Performance of Various Recognition Methods on Low-Quality Machine-Printed Digits

System	Filter	Assoc.	Trigger	Learn	Class	Speed
ART-1	external	corr.	max P	replace	Bayesian	7.4ms
ART-2	nonlinear	corr.	max P	average	Bayesian	≈1400ms
CORT-X	ART-2 & RF	corr.	max P	average	Bayesian	≈1400ms
FAUST	external 4 types	5 types	Assoc. Space	6 types of RF	Assoc. Space	2.5ms

of low-quality machine-printed digits with 100% recognition. The learning times for the ART-1 and FAUST methods are comparable to the recognition times. The learning times for the ART-2 and CORT-X methods are approximately ten times the recognition times. All of the methods are unconditionally stable and always converge. The advantage of the FAUST method over ART-2-based method is obtained by using Gabor filtering methods instead of a nonlinear filter. The advantage over ART-1-based methods is that the filter can be eliminated for simple recognition tasks, such as machine print, because edge and stroke variations are learned by the associative relevance memory.

The effectiveness of the FAUST architecture is further demonstrated on a character-recognition example providing greater than 99.5% recognition accuracy on low- to medium-quality machine-print with no substitutional errors and a 2.4 ms recognition time and 88% recognition on hand print with a 2.3% substitutional error rate and a 30.9 ms recognition time. The hand-print recognition time includes the 13.6 ms used for Gabor filtering.

REFERENCES

[1] G.A. Carpenter, S. Grossberg, and C. Mehanian, "Invariant Recognition of Cluttered Scenes by a Self-Organizing ART Architecture: CORT-X Boundary Segmentation," *Neural Networks*, Vol. 2, No. 3, 1989, pp. 169–181.

[2] B.L. Pulito, T.R. Damarla, and S. Nariani, "A Two-Dimensional Shift Invarient Image Classification Neural Network Which Overcomes the Stability/Plasticty Delemma," *Proceedings of the IJCNN, II*, pp. 825–833, June 1990.

[3] C.L. Wilson, R.A. Wilkinson, & M.D. Garris, "Self-Organizing Neural Network Character Recognition on a Massively Parallel Computer," *Proceedings of the IJCNN, II*, pp. 325–329, June 1990.

[4] 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, No. 2, February 1987, pp. 54–115.

[5] K. Fukushima, "Neocognitron: A Self-Organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shifts in Position," *Biological Cybernetics*, Vol. 36, No. 3, March 1980, pp. 193–202.

[6] J.G. Daugman, "Complete Discrete 2-D Gabor Transform by Neural Networks for Image Analysis and Compression," *IEEE Trans. on Acoustics, Speech, and Signal Processing*, Vol. 36, No. 7, July 1988, pp. 1169–1179.

[7] R. Linsker, "Self-Organization in a Perceptual Network," *Computer*, Vol. 21, No. 4, March 1988, pp. 105–117.

[8] J. Rubner and K. Schulten, "Development of Feature Detectors by Self-Organization," *Biological Cybernetics*, Vol. 62, No. 2, February 1990, pp. 193–199.

[9] A. Rojer and E. Schwatz, "Multimap Model for Pattern Classification," *Neural Computation*, Vol. 1, No. 2, 1989, pp. 104–115.

[10] D.L. Alkon, K.T. Blackwell, G.S. Barbour, A.K.Rigler, and T.P. Vogl, "Pattern-Recognition by an Artificial Network Derived from Biological Neuronal Systems," *Biological Cybernetics*, Vol. 62, No. 5, May 1990, pp. 363–376.

[11] K.G. Beauchamp, *Walsh Functions and Their Applications*, Academic, London, 1975, pp. 41–65.

[12] T. Kohonen, *Self-Organization and Associative Memory*, Second Edition, Springer-Verlag, Berlin, 1988, pp. 59–63.

[13] D.O. Hebb, *The Organization of Behavior*, Wiley, New York, 1949.

[14] M.A. Cohen and S. Grossberg, "Stability of Global Pattern Formation and Parallel Memory Storage by Competitive Neural Networks," *IEEE Trans. on Sys., Man and Cyber.*, Vol. 3, No. 5, Sept/Oct 1983, pp. 815–826.

[15] A.S. Householder, *The Theory of Matrices in Numerical Analysis*, Dover, New York, 1964, pp. 91–146.

[16] C.L. Wilson and M.D. Garris, "Hand-printed Character Database," *NIST Special Database 1*, National Inst. of Std. Tech, Gaithersburg, Md, HWDB, April 18, 1990.

[17] P.M.Flanders, D.J. Hunt, S.F. Reddaway, and D. Parkinson, "Efficient High Speed Computing with the Distributed Array Processor," *Proceedings of Symposium on High Speed Computer and Algorithm Organization*, University of Ill., 1977, pp. 113–128.

[18] C.L. Wilson, R.A. Wilkinson, and M.D. Garris, "Self-Organizing Neural Network Character Recognition Using Adaptive Filtering and Feature Extraction," *Progress in Neural Networks, Volume 3*, Ablex, Norwood, NJ, 1993 (in press).