12-1990

Analyzing Images Containing Multiple Sparse Patterns with Neural Networks

Rangachari Anand
Syracuse University

Kishan Mehrotra
Syracuse University, mehrtra@syr.edu

Chilukuri K. Mohan
Syracuse University, ckmohan@syr.edu

Sanjay Ranka
Syracuse University

SU-CIS-90-42

Analyzing Images Containing Multiple Sparse Patterns with Neural Networks

R. Anand, K. Mehrotra, C. Mohan, and S. Ranka

December 1990

School of Computer and Information Science
Syracuse University
Suite 4-116, Center for Science and Technology
Syracuse, New York 13244-4100

Analyzing Images Containing Multiple Sparse Patterns with Neural Networks

R. Anand
Kishan Mehrotra
Chilukuri K. Mohan
Sanjay Ranka

4-116 Center for Science and Technology
School of Computer and Information Science
Syracuse University
Syracuse, NY 13244-4100
U.S.A.

anand/kishan/mohan/ranka@top.cis.syr.edu
Phone: (315) 443-2368

Abstract

We have addressed the problem of analyzing images containing multiple sparse overlapped patterns. This problem arises naturally when analyzing the composition of organic macromolecules using data gathered from their NMR spectra. Using a neural network approach, we have obtained excellent results in using NMR data to analyze the presence of amino acids in protein molecules. We have achieved high correct classification percentages (about 87%) for images containing as many as five substantially distorted overlapping patterns.

SU-CIS-90-42

Analyzing Images Containing Multiple Sparse Patterns with Neural Networks

R. Anand, Kishan Mehrotra, Chilukuri K. Mohan, Sanjay Ranka

December 1990

School of Computer and Information Science
Suite 4-116
Center for Science and Technology
Syracuse, New York 13244-4100

(315) 443-2368

1 Introduction

The ability to analyze complex images containing multiple patterns is a difficult but important task in several real-world applications. A robotic vision system, for example, may be required to identify juxtaposed objects on a conveyer belt. In hand-written character recognition, an image may contain several, possibly overlapped, characters each of which must be recognized separately. The conventional approach in such problems is to segment the input image and then perform feature-extraction followed by classification [3]. This method is appropriate when the expected patterns consist of contiguous sets of distinct features. Special techniques are required for efficient analysis when images are large but sparse.

In this paper, we are concerned with the problem of analyzing images containing large multiple overlapped *sparse* patterns. Such patterns consist of a small number of features dispersed widely in the image. The features are usually small in size: perhaps just a single pixel. This classification problem is encountered when analyzing images obtained by certain types of NMR (Nuclear Magnetic Resonance) spectroscopy.

In the application described in this paper, a pattern that belongs to a single class will typically consist of about ten non-zero pixels in a 256×256 image. For each class of patterns, the features occur only in certain characteristic regions of the image. An image presented for classification will generally contain a number of patterns that belong to several different classes. Our task is to determine the classes to which we may attribute patterns contained in an input image.

Relatively little work has been reported in the literature on efficient neural network techniques for the analysis of images containing multiple patterns. One possible approach is to use Strong and Whitehead's physiological model [16] which describes how humans are able to pay selective attention to each pattern contained in a complex image in a sequential order. Fukushima's selective-attention neural network for classifying overlapped patterns [6] (based on the Neocognitron [5] [4]) gives some of the few successful results in this area. If a composite pattern consisting of several hand-written characters is presented, the network attends to those characters one at a time and recognizes individual characters.

The main problem in applying Fukushima's approach for large images task is the sheer size of the required network. As many as 41000 cells are needed for classifying

patterns in a 19×19 image. Since we must process considerably larger 256×256) images, the computational requirements using Fukushima's model are too high.

Our approach is to perform efficient analysis by exploiting the sparseness of images. We have developed a modular analyzer for the problem of analyzing images containing multiple sparse patterns. Each module detects the presence of patterns that belong to one class in the input image. Each module has two stages. The first stage is a feature detector based on clustering [8]. For each class of patterns, cluster analysis is used to identify those regions of the input image where the features of patterns belonging to that class are most likely to be found. The second stage of each module is a standard backpropagation-trained feed-forward neural network [15] which performs the tasks of thresholding and classification.

We have used our analyzer to analyze images obtained from NMR spectroscopy of proteins. Proteins are large complex molecules made up of building blocks called amino acids, of which there are 18 commonly occurring types. The presence of a constituent amino acid in a protein can be detected by observing a characteristic pattern in the NMR spectrum of the protein. We have been able to analyze correctly artificially generated NMR spectra of small proteins containing upto five different types of amino acids.

In the next section, we discuss the problem of analyzing multiple sparse patterns, describe some details of the NMR analysis problem, and discuss previous work on this topic. In section 3, we describe details of the analyzer. Details of the experiments and experimental results are presented in section 4. Section 5 contains concluding remarks.

2 The problem

Our problem is to analyze images containing multiple sparse patterns. Three sparse patterns all of which belong to the same class are shown in figure 1. Each pattern belonging to this class has three 'features'. The location of each feature (a feature is just single non-zero pixel) is shown by a '.' sign. As suggested by figure 1, the locations of the features are not fixed. However, they do remain within the feature-regions, bounded in figure 1 by dashed lines.

An overview of the classification process is depicted in figure 2. In the example

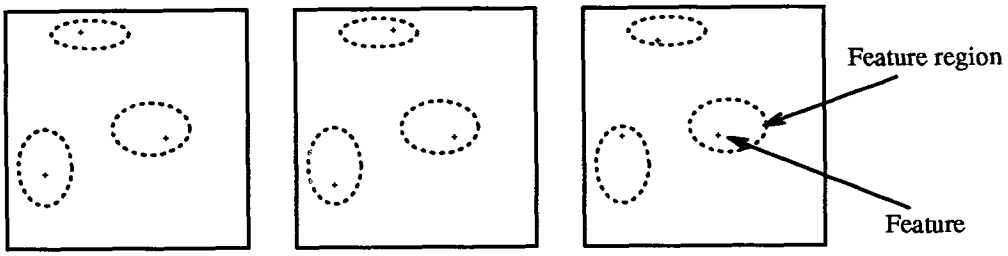

Figure 1: Three sparse patterns which all belong to one class

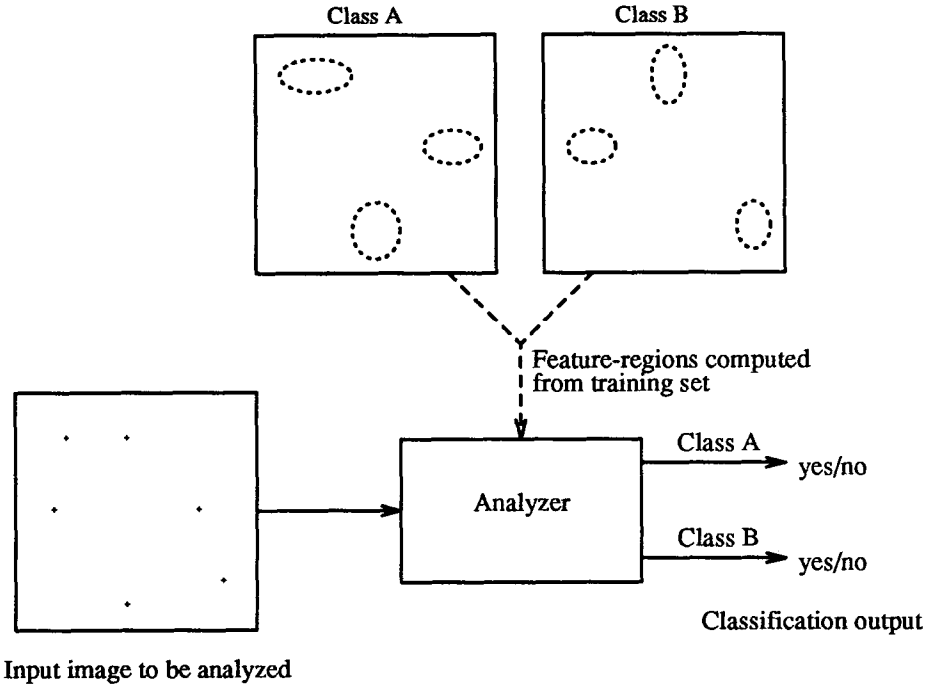

Figure 2: Overview of the classification process.

shown, the feature-regions for two classes are known to the analyzer. Using this knowledge, the analyzer can determine whether a pattern of either class is present in the input image.

If the feature-regions for different classes do not overlap, pattern recognition would be a simple task, since it would suffice to recognize positively any one feature of each class. In the applications that we are interested in, feature-regions for different classes *do* overlap. Consequently, a feature may lie within feature-regions of several classes. Although such a feature does not permit us to decide unambiguously whether a particular class of patterns is present in the image, it does however partially constrain

the classification. Overall, the problem of classifying images containing sparse patterns may be characterized as having multiple simultaneous constraints. Rumelhart and McClelland [15] note that such problems are ideal candidates for neural-network solutions.

2.1 Problem definition

In this section, we formally define the problem of analyzing images with multiple patterns. We first define our representation of an input image and then specify the information needed to characterize each class of patterns. We then describe the process by which this information is used to determine whether a particular class of patterns is present in an image.

Image representation: An input image is a two-dimensional array of non-negative integers called 'intensities'. Although images are quite large in practice, only a small number of non-zero elements are encountered in applications like NMR. For compactness, we will represent an image by a set of triples, where the first two components of each triple contain the location and the third is the intensity of a non-zero element in the input image. Chemists refer to each such triple as a *peak*. An image P is a set of peaks:

$$P = \{P_1, P_2, \ldots, P_N\}$$

where each $P_i = (P_{i_x}, P_{i_y}, P_{i_z})$ for $1 \leq i \leq N$.

An image, P, may contain several patterns, (disjoint subsets of P) where by 'pattern' we mean a collection of peaks associated with a certain class. We do not know in advance how many patterns are contained in an input image. Therefore, the number of peaks, N, in each image varies.

Characterizing a class of patterns: In some cases, we find that patterns which belong to one class may occur in several different configurations. We therefore define a set of *pattern-templates*, T_C, for each class C, where each pattern-template characterizes one configuration:

$$T_C = \{t_{(C,1)}, t_{(C,2)}, \ldots, t_{(C,M_C)}\}.$$

Each pattern-template, is a set of *feature-templates*:

$$t_{(C,j)} = \{F_{(C,j,1)}, F_{(C,j,2)}, \ldots, F_{(C,j,S_{(C,j)})}\}.$$

A feature-template $F_{(C,j,k)}$ contains a complete specification for a feature which could occur in a pattern belonging to class C. Feature-templates determine which features (peaks) are in fact present in an input image.

A feature-template $F_{(C,j,k)}$ consists of the expected location, $\mathbf{r}_{(C,j,k)}$, of a peak in the image, and a threshold $\lambda_{(C,j,k)}$:

$$F_{(C,j,k)} = (\mathbf{r}_{(C,j,k)}, \lambda_{(C,j,k)}).$$

$\mathbf{r}_{(C,j,k)}$ is the center of a feature region while $\lambda_{(C,j,k)}$ is used to define how far the feature region extends around the center. As we shall see in section 3, we obtain the values of $\mathbf{r}$ by cluster analysis. However, we do not compute the values of λ explicitly; it is determined implicitly when a neural network is trained.

Matching a feature-template: This is the first step in pattern recognition. We must determine whether a peak in the input image matches a feature-template. Given a peak P_i which may correspond to feature $F_{(C,j,k)}$, we define a measure of the distance between the two by a matching function. It is given by

$$\mu(P_i, \mathbf{r}_{(C,j,k)}) = P_{i_z}\, g(|\mathbf{r}_{(C,j,k)} - (P_{i_x}, P_{i_y})|)$$

where g, the 'error function', is chosen to decrease with distance $|\mathbf{r} - (P_{i_x}, P_{i_y})|$.

We say that a peak P_i 'matches' a feature-template $F_{(C,j,k)}$ if:

$$\mu(P_i, \mathbf{r}_{(C,j,k)}) > \lambda_{(C,j,k)}$$

Matching a pattern-template: We say that an input image P 'matches' a pattern-template $t_{(C,j)}$ if for each feature-template $F_{(C,j,k)} \in t_{(C,j)}$, we can identify a unique peak $P_i \in P$ such that P_i matches $F_{(C,j,k)}$.

Classification: If an input image P matches a pattern-template $t_{(C,j)}$, a pattern of class C is defined to be present in the input image. The overall analysis task is to determine all the classes whose features are present in the input image, hence the above procedure is repeated for every class.

2.2 Classification of NMR spectra

NMR (Nuclear Magnetic Resonance) spectroscopy is a powerful method for the determination of the three-dimensional structure of complex organic macromolecules such as proteins [18]. Proteins are long chains of smaller molecules called amino acids. Approximately 18 different types of amino acids are commonly found in proteins. The first step in analyzing the structure of a protein is to determine its constituent amino acids. One type of NMR spectroscopy used for this purpose is called Correlational spectroscopy (usually referred to as COSY).

A COSY spectrum is a two-dimensional integer array which is symmetric about the SW-NE diagonal. The spectra of the amino acids are sparse patterns of the kind described in section 2.1. The number of peaks in the spectrum of an amino acid is generally quite small (usually between 2 and 10). There is a slight variability in the positions of the peaks of an amino acid, which arises from interactions with the neighboring amino acids in the protein.

The spectrum of a protein is the result of the combination of the spectra of its constituent amino acids. The task of determining the constituent amino acids of a protein is therefore analogous to the task of analyzing an image containing multiple sparse patterns. The training set for our analyzer consists of a number of sample spectra for each type of amino acid. These spectra were generated from information about the distributions of peaks for each type of amino acid tabulated in [7].

Although we have only analyzed images obtained from one type of spectroscopy (COSY), there is actually a wide variety of NMR techniques available to the chemist. Most of the automatic peak-assignment programs described in the chemical literature use several different types of spectra as inputs. We expect the performance of our system to improve with the use of multiple sources of inputs.

Kleywegt et al. [13] have described a system called CLAIRE which takes as input the 'HOHAHA' (Homonuclear Hartmann-Hahn Spectroscopy) spectrum and the 'NOESY' (Nuclear Overhauser Effect Spectroscopy) spectrum of the protein being investigated. When used on a protein with 46 amino acids, it was able to identify the peaks for 34 of them correctly.

Another system devised by Eads and Kuntz [2] requires for input the COSY, the NOESY and HOHAHA spectra. For a protein with 51 amino acids, their program

was able to identify the peaks for 21 of them. The remaining peaks in the spectra were assigned manually. Although their program took only 15 minutes to run, the manual assignment required two weeks of effort.

The use of neural networks for this problem does not seem to have been investigated previously except in a rudimentary manner for classifying one-dimensional NMR spectra [17]

3 System description

In this section, we describe a modular analyzer for the analysis of images containing multiple sparse patterns. An important aspect of our system is the use of a clustering algorithm to train feature detectors for each class. In machine vision systems, clustering is often used for image-segmentation [14]. Note that we distinguish between the use of the term 'feature detection' in the context of statistical pattern recognition and in our system. We are using feature detection to identify *spatial* features in the patterns. In statistical pattern recognition, feature detection refers to the process of choosing the best set of variables to characterize a set of items [1].

We first describe how clustering is relevant to our problem and then present an overview of the analyzer. We then describe the internal details of the modules.

3.1 Clustering

Clustering is used to find the expected locations of features. We illustrate this with an example. Let us suppose that the training set for some class say, C, consists of the three images in figure 1. Each image contains one pattern, which belongs to class C. These images have been superimposed in figure 3. Clearly, the features occur in three clusters. The center of each cluster is the expected location of a feature.

The procedure may be summarized thus: for each class C, create a set R_C containing the locations of all features in an image created by superimposing all the training set images for class C. By applying a clustering algorithm to R_C, we determine the expected location of each feature.

We have investigated two clustering algorithms: K-means clustering algorithm [8] and the LVQ (Learning Vector Quantizer) [9]. We have found that the LVQ performs

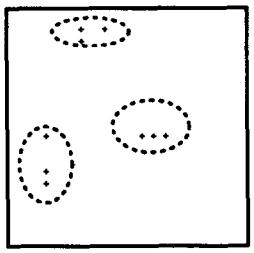

Figure 3: Use of clustering to find the expected locations of features.

better for our problem. A benchmarking study by Kohonen et. al. [11] also found that the LVQ produced better results than K-means clustering on several classification tasks.

The LVQ is a nearest-neighbor classifier which is trained by a competitive learning procedure. An LVQ is an array of units, each of which possess a reference vector m of the same dimensionality as the pattern space. In a trained LVQ, the reference vectors are the positions of the cluster centers. When a pattern is presented to the LVQ, only the unit whose reference vector is closest to the input pattern responds. Thus, the LVQ divides the pattern space into a number of regions, with one reference vector in each region. These regions are a Voronoi tesselation of the input space.

An additional property of the LVQ, which we do not use in our system, is that the training procedure creates an *ordered map* of the pattern space. In other words, if two units are close together in the array, their decision regions will be close to each other in the pattern space. Although LVQs with two-dimensional arrays of units are have been used for applications such as speech processing [12], we have obtained good results with a one-dimensional LVQ for our problem.

The learning procedure is now summarized briefly. Initially, the reference vectors are randomly distributed in the pattern space. Each training step requires the presentation of a pattern chosen randomly from the training set. A unit whose reference vector is the closest to the presented pattern is called the winner. The reference vectors of the winning unit *and its neighbors* in the array are adjusted so as to move them closer to the presented pattern. The amount by which the reference vectors are adjusted is gradually decreased during the learning procedure.

For each class C, we use the LVQ learning procedure on the set R_C to obtain a set of reference vectors. Before we describe how the reference vectors are used in our

analyzer, we make two observations regarding the choice of number of units used in the LVQ:

1. For a class C, we should have as many units as there are features. Then, the LVQ learning procedure will result in each unit being assigned to one cluster with their reference vectors being set to the respective cluster centers. When more units are used, the LVQ learning algorithm assigns more units to each cluster as a result of the ordered mapping property of units. Within a cluster, the units are distributed evenly. We take advantage of this property in the clustering filter.

2. We have observed that features sometimes occur very close together in patterns for the NMR problem. That is, the feature-regions for two features of a class may overlap substantially. In such cases, the LVQ cannot separate the two clusters, hence we need fewer units.

3.2 Overview of the analyzer

A block diagram of the analyzer is shown in figure 4. Each module detects the presence of one class of patterns. The modules work in parallel on an input image (presented as a list of peaks). Each module has two stages connected in series. The first stage, which extracts features from the input image, is called a clustering filter. The output of the first stage is a real vector called the feature vector. The second stage is a perceptron-like feed-forward neural network. Parts of the pattern recognition process defined in section 2.1 are performed in both stages. The clustering filter computes the values of the matching functions, while the thresholding is done in the neural network. Both stages are described in detail below.

3.3 The clustering filter

The role of each clustering filter (shown in figure 5) is to extract relevant information from the input image for one class of patterns. A clustering filter consists of a number of *feature detectors*. A feature detector is a processing unit which is activated by the presence of a feature (peak) in an associated region of the input image called the *receptive field*. The output of a feature detector is a real value which depends on the

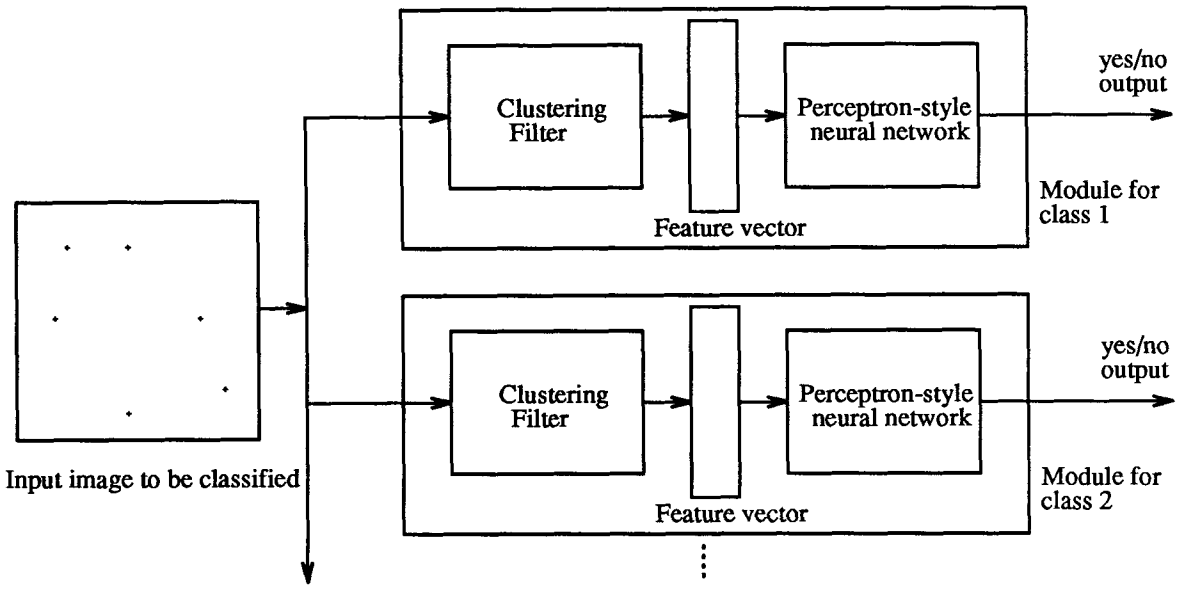

Figure 4: Block diagram of part of the modular analyzer.

number of peaks present within its receptive field, the intensities of those peaks and their distances from the center of the receptive field. The output of a clustering filter is a real vector, called the feature vector, each element of which is the output from one feature detector in the filter.

In a filter for class C, the receptive fields of the feature detectors should coincide with the feature regions of class C. For simplicity, we use feature detectors with *fixed* size receptive fields in our system. Consequently, if a feature region is larger than the receptive field, several feature detectors are required to cover it. We use the LVQ learning procedure to obtain the positions for the feature detectors.

The output from a feature detector located at coordinate $\mathbf{r}$, when presented with an image $P = \{P_1, P_2, \ldots, P_N\}$ is shown below:

$$Output(\mathbf{r}) = \sum_{i=1}^{N} \mu(P_i, \mathbf{r})$$

The kernel function chosen is $g(x) = \exp^{-\tau x^2}$ where τ is a constant between 0.1 and 0.5. Although all peaks in the image are fed to the feature detector, it will actually respond only to those peaks which are very close to $\mathbf{r}$. This is because for the values of τ that we have chosen, the kernel function $g(x)$ drops to almost zero when $x > 6$. Therefore the feature detector only responds to peaks which lie within

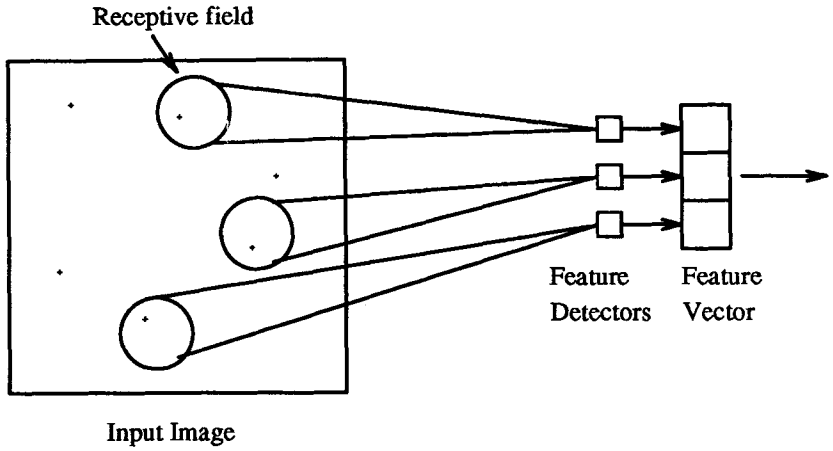

Figure 5: Clustering filter

a radius of 6 pixels around its center, r.

As we noted previously, there are cases where two peaks sometimes occur very close together. We do not need to make any special provision for this situation. We use only one set of feature detectors to cover the *combined* feature region. The output from these feature detectors will be higher, but this is easily handled by the neural network.

3.4 The neural network

Although a clustering filter is trained to respond most strongly to patterns of a particular class, it is possible (due to overlap of feature-regions) that some of the detectors of one class may be activated when patterns of another class are presented. We use a neural network to determine implicitly the appropriate thresholds for each pattern detector. A neural network is trained after the clustering filters of the first stage have been trained and set up.

For each class C, the neural network (of the corresponding module) must be taught to discriminate between feature vectors obtained from images containing a pattern of class C and feature vectors produced from images which do not contain patterns of class C.

We use backpropagation [15] to train the network. Backpropagation is a supervised learning algorithm in which a set of patterns to be learnt are repeatedly presented to the network together with their target output patterns. At the outset

of a training session, the weights and biases are randomly initialized. For each pattern presented, the error backpropagation rule defines a correction to the weights and thresholds to minimize the square sum of the differences between the target and the actual outputs. The learning process is repeated until the average difference between the target and the actual output falls below an operator-specified level.

4 Results

In this section, we describe our experiments in training and testing the sparse image recognition system, and report the results obtained.

4.1 System parameters

To substantiate our approach, seven modules were trained for the NMR protein analysis problem. Each module can detect patterns corresponding to one amino acid. The final output from each module is a yes/no answer about whether the respective class (amino acid) is judged to be present. From among 18 possible amino acids, we trained modules for seven amino acids whose spectra appeared to be the most complex, with more peaks than the others. But in training as well as testing the modules, we used data which included peaks from the other 11 amino acids as well, and obtained good results in analyzing the presence of the seven amino acids for which modules were trained. This shows that an incremental approach is possible: upon building modules for the other 11 amino acids, we expect that our results will continue to hold.

Table 1a lists the parameters of each module. The names of the amino acids along with their one letter codes are listed in the first column. The second column is the number of feature detectors in the first stage of each module. The third column shows the number of hidden layer nodes in the neural network which comprises the second stage of each module. These were approximately the smallest number of nodes required for convergence of the network training procedure, obtained by experimenting with various values.

Module	Number of detectors	Hidden layer size
Alanine (a)	4	10
Cystine (c)	10	20
Aspartic acid (d)	19	20
Glutamic acid (e)	14	10
Phenylalanine (f)	26	10
Isoleucine (i)	19	20
Valine (v)	15	20

Table 1a: Module Parameters.

The training set consists of a total of 90 single-class images, with 5 for each of the 18 amino acids. The equation indicating how weights are changed using the error back-propagation procedure [15] is:

$$\Delta w_{ji}(n+1) = \eta(\delta_p j o_p j) + \alpha \Delta w_{ji}(n) \tag{1}$$

In each module, we used a value of $\eta = 0.1$ for the learning rate parameter, and a value of $\alpha = 0.05$ for the momentum coefficient. The target mean squared error (to terminate the network training procedure) was set to be 0.01. During training, the target output for the networks was set to be 0.1 when the required answer was 'no' and 0.9 when the required answer was 'yes'. Weights in the network were updated only at the end of each 'epoch' (one sequence of presentations of all training inputs). Table 1b shows the number of epochs needed to train the LVQ's and the feedforward (backpropagation) neural networks, for each module.

Module	LVQ	Backpropagation
a	10000	717
c	10000	2653
d	20000	452
e	20000	149
f	30000	3128
i	20000	503
v	20000	328

Table 1b: Number of epochs required for training.

To investigate the effect of varying the receptive field sizes of detectors, we trained two versions of each module with different values for τ, the constant in the kernel function. When $\tau = 0.5$, detectors have small receptive fields, whereas when $\tau = 0.1$, detectors have large receptive fields.

4.2 Experimental results

The goal of the experiments was to measure the correctness of overall classification when the system was presented with composite images containing several patterns of different classes. We now describe the various experiments performed to test our sparse image analysis system on composite images consisting of:
(i) different numbers of patterns;
(ii) with and without perturbations; and
(iii) for detectors with different receptive fields ($\tau = 0.5$ and $\tau = 0.1$).

To illustrate the testing method, consider a composite image created by superimposing two images with patterns that belong to classes *c* and *d*. Correct classification implies that this image should be classified 'NO' by modules *a, e, i, f* and *v* and 'YES' by modules *c* and *d*. We measure the percentages of correct classification, testing the modules in this manner on various composite images.

In the *first* set of experiments, we generated 1000 examples of each case: composite images containing 2, 3, 4 and 5 patterns respectively. In each set, the composite images were created by superimposing a randomly chosen set of images (each of a different class) drawn from the training set. The percentages of these images correctly classified by each module under different conditions are reported in table 2, for $\tau = 0.5$ and $\tau = 0.1$.

From table 2, it is clear that error rates increase with the number of images (patterns) in the input image. This is because the receptive fields of different classes of patterns overlap. Hence patterns of one class may partially activate feature detectors for other classes. As the number of patterns in the input image increases, it becomes increasingly likely that a feature detector may respond to artifacts arising from a fortuitous combination of patterns belonging to other classes. This problem is further aggravated by increasing the size of the receptive fields, as shown in table 2.

Module	$\tau = 0.5$				$\tau = 0.1$			
	No. of patterns in image				No. of patterns in image			
	2	3	4	5	2	3	4	5
a	100	100	100	100	99.1	99.3	98.9	97.7
c	100	99.8	99.9	99.7	100	100	100	100
d	100	100	99.8	99.5	100	99.9	99.1	98.5
e	100	100	100	100	96.4	94.4	90.3	86.3
f	99.9	99.5	99.3	98.5	99.3	95.1	91.0	86.4
i	98.0	96.4	96.4	95.7	99.9	99.6	99.6	99.3
v	100	100	100	99.6	99.4	99.2	98.7	94.4

Table 2: Percentages correctly classified, when test images were random combinations of training set images.

Module	$\tau = 0.5$				$\tau = 0.1$			
	No. of patterns in image				No. of patterns in image			
	2	3	4	5	2	3	4	5
a	98.0	97.8	97.8	95.8	99.7	99.6	98.6	96.6
c	97.1	95.0	95.0	94.0	99.7	99.3	99.0	99.2
d	98.6	97.1	95.5	94.8	100	99.6	99.4	98.2
e	99.4	99.2	98.6	98.3	97.0	95.0	88.6	87.7
f	96.1	93.2	93.6	90.1	98.9	97.0	89.8	84.6
i	96.4	94.6	93.7	92.6	99.7	99.6	99.7	99.1
v	98.3	97.0	96.4	95.1	99.8	99.4	96.7	95.5

Table 3: Percentages correctly classified, with low noise. Test images were random combinations of training set images with peak locations randomly translated by [-1, +1].

Module	$\tau = 0.5$				$\tau = 0.1$			
	No. of patterns in image				No. of patterns in image			
	2	3	4	5	2	3	4	5
a	92.0	85.8	82.5	80.7	94.6	90.9	86.8	86.1
c	94.0	91.0	89.0	83.7	95.3	94.0	92.7	89.7
d	89.2	85.7	79.0	76.4	95.2	94.6	92.0	89.8
e	89.4	83.8	79.0	76.2	96.6	95.2	91.5	86.8
f	87.4	82.2	75.8	70.6	94.4	89.8	84.8	76.8
i	89.8	82.8	77.5	74.4	95.5	93.6	93.2	89.7
v	90.0	85.8	80.8	75.5	94.1	91.3	87.7	88.6

Table 4: Percentages correctly classified, with high noise. Test images were random combinations of training set images with peak locations randomly translated by [-3, +3].

It is desirable to perform correct classification even in the presence of small errors or corrupted data. Hence, we tested our system with composite images produced by superimposing *distorted* versions of the training set images to the system.

Two series of experiments were performed, varying the amount of distortion in the test data, for small and large receptive fields ($\tau = 0.5$ and $\tau = 0.1$). In one set of experiments, distortions were introduced by adding a random integer in the range [-1, +1] to the coordinates of the peaks. The results of these are summarized in table 3. In another set of experiments, the distortion was increased by adding random integers in the range [-3, +3] to the coordinates of peaks. These results are summarized in table 4.

With the small receptive field system ($\tau = 0.5$), the combined effect of distortion and multiple patterns causes classification accuracy to deteriorate substantially. On the other hand, classification capabilities of the large receptive field system ($\tau = 0.1$) are less affected and degrade more gracefully with noise. This phenomenon contrasts with the observation that the small receptive field system performs marginally better on uncorrupted test data.

5 Concluding Remarks

In this paper, we have addressed the problem of analyzing images containing multiple sparse overlapped patterns. This problem arises naturally when analyzing the composition of organic macromolecules using data gathered from their NMR spectra. Using a neural network approach, we have obtained excellent results in using NMR data to analyze the presence of amino acids in protein molecules. We have achieved high correct classification percentages (about 87%) for images containing as many as five substantially distorted overlapping patterns.

The architecture of our system is modular: each module analyzes the input image and delivers a yes/no output regarding the presence of one class of patterns in the image. Each module contains two stages: a clustering filter, and a feedforward neural network. An unconventional aspect of our approach is the use of clustering to detect *spatial* features of patterns. This may be compared to the use of clustering for image-segmentation in several machine vision systems.

We performed a number of experiments to measure the correctness of overall classification when the system was presented with composite images containing several patterns of different classes. We tried two versions of the system, one with small receptive field detectors and the other with large receptive field detectors. In both cases, we observed that the rate of correct classification decreased as the number of patterns in the image was increased. To determine the ability of the system to cope with variations in the patterns, images with random perturbations to the patterns were presented to the system in another series of experiments. In this case, we observed that the classification abilities of the large receptive field system are less affected and degrade more gracefully.

The classification process described in this paper is only the first step in the analysis of NMR spectra. It is of considerable interest to chemists to determine the precise association of the peaks in in the input image with different patterns. We are currently working on an extension to the system described in this paper to perform this task. We plan to refine the clustering algorithm to enable the use of feature-detectors with variable size receptive fields. We expect to improve performance by combining the evidence from multiple input sources, as is done in other NMR analysis methods.

Acknowledgments

We gratefully acknowledge the assistance received from Sandor Szalma and Istvan Pelczer of the NMR Data Processing Laboratory, Chemistry Department, Syracuse University. The images for the training set were generated by a program written by Sandor Szalma.

References

[1] Chen, C. "Statistical Pattern Recognition". Spartan Books, 1973.

[2] Eads, C. D., and Kuntz, I. D. "Programs for Computer-Assisted Sequential Assignment of Proteins". *Journal of Magnetic Resonance*, vol. 82, pp 467-482, 1989.

[3] Fu, K. S. and Mui, J. K. "A Survey on Image Segmentation". *Pattern Recognition*, vol. 13, pp. 3-16, 1981.

[4] Fukushima, K. "Neocognitron: A Self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position". *Biological Cybernetics*, vol. 36, pp. 193-202, 1980.

[5] Fukushima, K., Miyake, S., and Ito, T. "Neocognitron: A Neural Network Model for a Mechanism of Visual Pattern Recognition". *IEEE Trans. on Sys., Man and Cyber.*, **SMC-13**, pp. 826-834, 1983.

[6] Fukushima, K. "Neural Network Model for Selective Attention in Visual Pattern Recognition and Associative Recall". *Applied Optics*, vol. 26, no. 23, pp. 4985-4992, 1987.

[7] Gross, K-H., Kalbitzer, H. R. "Distribution of Chemical Shifts in 1H Nuclear Magnetic Resonance Spectra of Proteins". *Journal of Magnetic Resonance*, vol. 76, pp 87-99, 1988.

[8] Jain, A. K. and Dubes, R. C. "Algorithms for Clustering Data". Prentice Hall, 1988.

[9] Kohonen, T. "Self-organized Formation of Topologically Correct Feature Maps". *Biological Cybernetics*, vol. 43, pp. 59-69, 1982.

[10] Kohonen, T. "Analysis of a Simple Self-Organizing Process". " *Biological Cybernetics*, vol. 44, pp 135-140, 1982.

[11] Kohonen, T., Barnes, G., and Chrisley, R. "Statistical Pattern Recognition with Neural Networks: Benchmarking Studies". *Proc. of the IEEE Int. Conf. on Neural Networks*, vol. 1, pp. 61-68, 1988.

[12] Kohonen, T. " The "The Neural" Phonetic Typewrite". *IEEE Computer*, pp. 11-22, March 1988.

[13] Kleywegt, G. J., Lamerichs, R. M. J. N., Boelens, R., and Kaptein, R. " Toward Automatic Assignment of Protein 1H NMR Spectra". *Journal of Magnetic Resonance*, vol. 85, pp. 186-197, 1989.

[14] Nagao, M. and Matsuyama, T. "A Structural Analysis of Complex Aerial Photographs". Plenum Press, 1980.

[15] Rumelhart, D. E., and McClelland, J. L. "Parallel Distributed Processing, Volume 1". MIT Press, 1987.

[16] Strong, G. W. and Whitehead, B. A. "A Solution to the Tag-Assignment Problem for Neural Networks". *Behavioral and Brain.Sciences*, vol. 12, pp. 381-433, 1989.

[17] Thomsen, J. U., and Meyer B. "Pattern Recognition of the 1H NMR Spectra of Sugar Alditols Using a Neural Network". *Journal of Magnetic Resonance*, vol. 84, pp. 212-217, 1989.

[18] Wuthrich, K. "NMR of Proteins and Nucleic Acids". John Wiley & Sons, New York, 1986.