Texture classification using ART-based neural networks and Fractals

Dimitrios Charalampidis, Takis Kasparis, Michael Georgiopoulos

Department of Electrical and Computer Engineering
University of Central Florida
Orlando, Florida, 32816

ABSTRACT

In this paper texture classification is studied based on the fractal dimension (FD) of filtered versions of the image and the Fuzzy ART Map neural network (FAMNN). FD is used because it has shown good tolerance to some image transformations. We implemented a variation of the testing phase of Fuzzy ARTMAP that exhibited superior performance than the standard Fuzzy ARTMAP and the 1-nearest neighbor (1-NN) in the presence of noise. The performance of the above techniques is tested with respect to segmentation of images that include more than one texture.

Keywords: Classification, texture, fractal dimension, Fuzzy ART Map, nearest neighbor.

1. INTRODUCTION

Texture classification is an important task in many applications in computer vision and pattern recognition. It is a process that involves classification of the feature vectors that are extracted from a textured image. Features are parameters that characterize the texture and feature vector is a set that consists of more than one feature. The selection of this set is very important because it must sufficiently identify different textures; otherwise the classification results will not be good independently of the classification algorithm. Classification usually involves training of a system so that it will be able to identify textures. A class can be defined as a collection of objects that have the same characteristics. Textures that belong to the same class are given the same label. The system is trained using feature vectors that correspond to textures with already known label. The performance of the algorithm can be tested by studying the classification for textures that were not used in the training process but they have known labels. Also, the performance can be tested by studying the classification for the training set when it is subjected to noise. Neural networks (NN) are often used for classification purposes, such as Back Propagation NN, Radial Basis Function NN, Fuzzy ART Map neural network (FAMNN)[1], etc. Other common classification methods such as stochastic models and Nearest Neighbor techniques are also used.

2. BACKGROUND

2.1 Fractal Dimension

There are many definitions of the FD of an object, including box dimension, intersection dimension and Bouligand-Minkowski dimension. FD has been characterized as a measure of irregularity of an object. Any curve is an object with one topological dimension that occupies some part of a surface. FD defines how much area of this surface is occupied by the curve. For instance, a highly irregular curve will have larger FD than a straight line. The FD of a curve can be between 1, which is equal to its topological dimension, and 2 which is equal to the topological dimension of the surface that it can occupy. The concept of FD can be extended to surfaces. The FD of a surface can be between 2, which is its topological dimension, and 3, which is the topological dimension of the "box" that the surface can occupy.

Variation[2] is a method that is used to compute the FD of an object. It has been shown that the variation method gives accurate and robust estimation of the FD of a surface. An image Z(x, y) of size R x R can be considered as a surface of size R x R, where its value at position (x_o, y_o) is $Z(x_o, y_o)$. The variation method can be applied on an image so that its FD can be computed.

The definition of the FD for surfaces using the variation method, is that if a surface Z is a fractal, there exists at least one part of the interval [0, 1] where Z is nowhere or almost nowhere differentiable. If P(x, y, x', y') is the slope of the line passing through points (x, y, Z(x, y)) and (x', y', Z(x', y')), then this |P(x, y, x', y')| goes to infinity as the point (x', y') tends toward (x, y). FD is defined as the rate in which |P(x, y, x', y')| goes to infinity. The variation of Z can be defined as:

$$V_\varepsilon(x, y) = \max_{\mathrm{dist}((x, y),(s, t)) \le \varepsilon} Z(s, t) - \min_{\mathrm{dist}((x, y),(s, t)) \le \varepsilon} Z(s, t) \qquad (1)$$

where dist((x, y),(s, t))=max(|x-s|, |y-t|) and $\varepsilon > 0$. The integral of $V_\varepsilon(x, y)$ tends to zero as ε tends to 0. The rate of growth of this integral is directly related to the FD of Z. The FD of the surface Z is given by:

$$FD_Z = \Delta_V(Z) = \lim_{\varepsilon \to 0} \left(3 - \frac{\log \int_0^1 \int_0^1 V_\varepsilon(x, y)dxdy}{\log \varepsilon} \right) \qquad (2)$$

$$= \lim_{\varepsilon \to 0} \left(\frac{\log \int_0^1 \int_0^1 [\, V_\varepsilon(x, y)/\varepsilon^3 \,]\, dxdy}{\log (1/\varepsilon)} \right) \qquad (3)$$

The slope of the log-log plot of the line that is defined by $\log \int_0^1 \int_0^1 [\, V_\varepsilon(x, y)/\varepsilon^3 \,]\, dxdy$ and $\log(1/\varepsilon)$ gives the FD of the surface. The computation of the FD of a discretized surface involves substitution of the integrals with summations.

The FD of an image can be computed locally in all different regions of size R x R of the image, so that a FD space can be created. This FD space will be mapped one-to-one to the pixels of the image. The algorithm for computing the FD space of an image is implemented as follows: The difference $V_{\varepsilon/R}$ between the maximum and the minimum grayscale values is computed in a small window of size T x T, where $T = 2\varepsilon+1$. This window is centered at the pixel with coordinates (x, y). This computation is repeated for all pixels (x, y) of the image, for $\varepsilon = 1, 2, 3, \ldots, \varepsilon_{max}$. $V_{\varepsilon/R}(x, y)$ is the ε^{th} variation located at (x, y). If we define $E_{\varepsilon/R}$ as the average of $V_{\varepsilon/R}(x, y)$ over a window W of size R x R, then the FD located at the window W is the slope of the line that better fits the points $(\log(R/\varepsilon), \log\{(R/\varepsilon)^3 E_{\varepsilon/R}\})$ where $\varepsilon = 1, 2, 3, \ldots, \varepsilon_{max}$. The line that better fits these points can be found using the least mean square approach. This FD is mapped to the central pixel of the window W. The next step is to shift the window W and map its FD to the central pixel of the new window. The previous steps are repeated for all pixels of the image and the FD space is created.

2.2 Gabor Filters

Multi-channel Gabor decomposition[4, 5, 6] is an approach to texture characterization and segmentation. The frequency spectrum of a signal, texture in this case, is decomposed into its spectral components using two dimensional Gabor filters with specified bandwidths and center frequencies. The filters can have constant or octave bandwidth. Two-dimensional directional symmetric Gabor filters can be defined in the spatial and the frequency domain respectively as:

$$h(x, y)=g(x', y')\,[\cos(2\pi\, f_o\, x') + j \sin(2\pi\, f_o\, x')] \qquad (4)$$

$$H(u, v)=4\pi\sigma_x\sigma_y \exp\{-[(u'-fo)^2/2\sigma_u^2+v'^2/2\sigma_v^2)]\} \tag{5}$$

where

$$g(x, y)=\exp\{-[x^2/2\sigma_x^2+y^2/2\sigma_y^2]\} \tag{6}$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{pmatrix} \cos\Phi o & \sin\Phi o \\ -\sin\Phi o & \cos\Phi o \end{pmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \tag{7}$$

$$\begin{bmatrix} u' \\ v' \end{bmatrix} = \begin{pmatrix} \cos\Phi o & \sin\Phi o \\ -\sin\Phi o & \cos\Phi o \end{pmatrix} \begin{bmatrix} v \\ u \end{bmatrix} \tag{8}$$

Here, g(x, y) is the Gaussian envelope, Φo denotes the orientation of the filter with respect to u axis, fo denotes the center frequency of the filter and $\sigma_u = 1/2\pi\sigma_x$, $\sigma_v = 1/2\pi\sigma_y$. It can be noticed that H(u, v) is also a Gaussian envelope centered at frequencies fo, 0. The parameters σ_x, σ_y are the standard deviations of the Gaussian envelope in the spatial domain and in the direction of x' and y' respectively and they define the size of the envelope. The standard deviations of the Gaussian envelope in the frequency domain and in the direction of u' and v' respectively are σ_u, σ_v.

Real-valued Gabor filters, are a special case of the complex-valued Gabor filters and they can be defined in the spatial and the frequency domain respectively as:

$$h(x, y)=g(x', y')\cos(2\pi\, uo\, x') \tag{9}$$

$$H(u, v)=2\pi\sigma_x\sigma_y\left(\exp\{-[(u'-uo)^2/2\sigma_u^2+v'^2/2\sigma_v^2)]\}+\exp\{-[(u'+uo)^2/2\sigma_u^2+v'^2/2\sigma_v^2)]\}\right) \tag{10}$$

2.3 Fuzzy ART Map Neural Network

The FAMNN consists of two Fuzzy ART modules designated as ART_a and ART_b, as well as an inter-ART module as shown in Figure 1. The inputs are presented at the ART_a module and the corresponding outputs to ART_b module. The inter-ART module determines whether the mapping between the inputs and the outputs is the correct one. In the case where the mapping between inputs and outputs is many to one, the FAMNN operations can be completely described by referring only to the ART_a module. For this reason only ART_a module is described here.

Some preprocessing of the input patterns takes place before they are presented to ART_a module. The first preprocessing stage takes as an input an M_a-dimensional input pattern and transforms it into an vector $\mathbf{a} = (a_1,\dots, a_{Ma})$, whose every component of **a** lies in the interval [0,1]. The second preprocessing stage accepts the vector **a** as an input and produces a vector **I**, such that

$$\mathbf{I} = (\mathbf{a}, \mathbf{a}^c) = (a_1,\dots,a_{Ma},a_1^c,\dots, a_{Ma}^c) \tag{11}$$

where

$$a_i^c = 1 - a_i,\ 1 \le i \le M_a \tag{12}$$

The above transformation is called complement coding and it is performed in ART_a at the preprocessor field F_o^a.

Every node i ($i \in \{1, 2, \ldots, 2M_a\}$) in F_1^a field is connected via a bottom-up weight with every node j ($j \in \{1, 2, \ldots, N_a\}$) in the F_2^a field; this weight is denoted $W^a ij$. Also, every node j in F_2^a field is connected via a top-down weight with every node i in the F_1^a field; this weight is denoted $w^a ji$. The vector whose components are equal to the top-down weights emanating from node j in the F_2^a field is designated by $\mathbf{w}^a_j = \{w^a_{j1}, \ldots, w^a_{j2Ma}\}$ and it is called ART_a template. The vector whose components are equal to the bottom-up weights converging to node j in the F_2^a field is designated by $\mathbf{W}^a_j = \{W^a_{j1}, \ldots, W^a_{j2Ma}\}$. FAMNN operates in two distinct phases: the training phase and the testing phase. These phases can be described by referring only to the top-down weights of the ART_a. Before discussing the two phases it is important to present the Fuzzy ARTMAP parameters of ART_a module.

1. β_a: This parameter is called the ART_a choice parameter, and takes values in the interval $(0, \infty)$. Its value affects the bottom-up inputs that are produced at the F_2^a nodes due to a pattern representation at F_1^a.
2. $\overline{\rho_a}$: This parameter, called the baseline vigilance parameter, determines the initial value of the vigilance parameter ρ_a in ART_a. The range of $\overline{\rho_a}$ is the interval [0,1]. Small values of $\overline{\rho_a}$ result in coarse clustering, while large values of $\overline{\rho_a}$ result in fine clustering of the input patterns presented in ART_a.
3. ρ_a: This parameter is called the vigilance parameter. Prior to an input/label pair presentation it is set equal to $\overline{\rho_a}$. During training, it is allowed to increase. It is set back to $\overline{\rho_a}$ when a new input pair is presented.
4. N_a: This parameter corresponds to the number of committed nodes +1 uncommitted in F_2^a during the training phase. A committed node is a node that has coded at least one input pattern. An uncommitted node is a node that is not committed.
5. ε: This parameter is used to evaluate the value of ρ_a when it is required to increase above $\overline{\rho_a}$. It is usually taken to be a very small positive constant.

The training phase works as follows: Given a list of training input/label pairs, such as $\{\mathbf{I}^1, L^1\}$, $\{\mathbf{I}^2, L^2\}$, $\{\mathbf{I}^{MP}, L^{MP}\}$, we want to train FAMNN to map every input pattern of the training list to its corresponding label. In order to achieve this goal, the training set is presented repeatedly to the architecture. The input patterns are presented to ART_a module. The training list is presented as many times as it is necessary for FAMNN to correct classify all the input patterns. The learning is complete when the weights do not change during a list presentation. This training scenario is called off-line learning. The testing phase works as follows: A test input pattern is mapped to label L, if L is the label of the node whose bottom-up input is the largest among all nodes in F_2^a.

The templates $\mathbf{w}_j^a$ that are formed at the F_2^a field during training are compressed representations of the training input patterns. Template $\mathbf{w}_j^a$ in F_2^a has an interesting geometrical interpretation. It can be presented by a hyper-box in the M_a-dimensional space. This hyper-box includes within its boundaries all the training input patterns that were coded by the template. A hyper-box can be defined by its endpoints. These are the endpoints of the hyper-box points with the smaller and larger coordinates of the hyper-box points. The first M_a elements of the template define the lower endpoint and the last M_a elements define the upper endpoint. The size of the hyper - box is defined by the vigilance parameter ρ_a. The larger the vigilance parameter the smaller will be the size of the hyper-box.

2.4 K-Nearest Neighbor (K-NN)

K-NN is a classification technique that uses directly the training data. A pattern belongs to class A if the majority of the K training patterns that are closer to it belong to this class. The closeness between two patterns is based on a distance measure. A very common distance measure is the Euclidean distance. 1-NN is a special case of K-NN for K = 1. In this case, a pattern belongs to class C if the training pattern that is closer to it, belongs to this class.

3. A CLASSIFICATION TECHNIQUE

3.1 Feature extraction

Before training takes place, the feature vectors for the training set must be extracted. The feature vectors consist of twelve FD-based features. FD has been used before for texture characterization and segmentation[6, 7]. The features that are used in this paper are the FD of the twelve filtered versions of the original image. The variation method is used to compute the FD. Directional non - symmetric Gabor filters are used for filtering for four directions (0^o, 45^o, 90^o, 135^o) and three central frequencies (0, 0.05, 0.1). The standard deviations are $\sigma_x = 6$ and $\sigma_y = 0.01$. In our previous work this set has been shown to be sufficiently good to characterize texture since it has good discriminatory power, it is independent of the image intensity and it is insensitive to multiplicative noise. Also, it has enough tolerance to noise.

Each FD feature is computed over a shifting window of size 16 x 16 for all filtered versions of the image. The computed feature is mapped to the central pixel of the window. Since each pixel of the filtered versions of the image can be mapped to the pixel of the original image with the same coordinates, the twelve features that are computed for each pixel of the filtered versions of the image can be mapped to the corresponding pixel of the original image. This feature extraction technique is repeated for all pixels so that W x W feature vectors can be computed where W is the width of the image.

3.2 Classification algorithm

An approach that uses FAMNN and takes into consideration that the image can be contaminated by noise is proposed. This approach takes into account the fact that the FD of an image tends to increase as the variance of the noise that it is subjected to, increases. This fact has been shown experimentally. This approach is based on the assumption that smaller FD values tend to increase more than larger FD values. This assumption seems reasonable since small FD means that the surface of the image is smooth. If this image is subjected to uncorrelated additive noise, then it is going to be rougher. If FD has a large value, then the surface of the image is already rough so that additive noise will not affect it so much. Of course there is also a possibility that the FD value of the noisy image will be smaller than the FD of the noise-free image and especially if FD has a very large value close to 3. Generally, it has been noticed that the FD for most textures does not have values very close to 3.

The FAMNN is used because it is a fast algorithm that converges in a small number of iterations. The training process is less time consuming than the training process of other common neural networks. The testing phase of FAMNN is less time consuming than nearest neighbor techniques since only a compressed version of the original training data information is used.

3.2.1 Training

The network is trained using 5120 feature vectors. These have been extracted from 20 images of size 256 x 256 by selecting the feature vector that is mapped to the pixel that is located at the center of a window of size 16 x 16. If all non-overlapping windows are considered, then we have a total of 256 feature vectors selected from each image. This sampling of the set of the feature vectors is necessary so that the size of the training set is decreased without loosing important information since the values of the features that correspond to pixels that are close, are similar especially after smoothing of the features has taken place. The total number of nodes that are created in F2 layer is 746.

For the training phase, features are smoothed by averaging them over a window so that their robustness to describe texture is increased. The smoothing window has size 33 x 33. The feature vector that corresponds to pixel with coordinates x, y is the smoothed feature vector over the window which is centered at x, y.

3.2.2 Testing

We have implemented a variation of the testing phase of FAMNN. The bottom-up input for every node in the F_2^a field is modified so that preference is given to nodes whose corresponding hyper-boxes have upper endpoints with smaller coordinates. Nodes that are created from feature vectors that have smaller elements are given more power since the values of these elements are expected to increase in the presence of noise, tending to be mapped to hyper-boxes with larger coordinates. The reason why only the coordinates of the upper endpoint of the hyper - box are considered is that the coordinates of the upper endpoint are indicative of the tolerance of this node to noise. If a feature vector is located in the feature space close to the lower endpoint then it has space to move that depends on the upper endpoint since it is expected to move to higher coordinates in presence of noise.

The modification is expressed as multiplication of the input with a term equal to $G(F) = M_a / (M_a + \alpha F)$, where M is equal to the number of elements of the feature vectors. F is the summation of the coordinates of the upper corner of the hyper-box that is defined by the node, and it is equal to:

$$F = \sum_{i=M_a+1}^{2M_a} (1 - w_{ji}^{a}) \tag{13}$$

and α is a constant that has the largest possible value, so that the correct classification of the training set will not be less than 99.5%. This value was found to be equal to 0.3. G(F) is a non-linear function of F and does not change very rapidly if F is large. This is desired since the FD for the pure noise surface might be large but not larger than a certain value. This value depends on the type of noise. If the FD of the noise-free signal was greater than the FD of the noise surface, then it would not be good to assume that the FD of the noisy image will be greater than the FD of the noise-free image. Since it is assumed that the type of the noise and its variance are not known it is appropriate not to affect very much the input to the F_2^a layer for the nodes with corresponding large hyper - box coordinates.

The testing phase works as follows: Initially, the features are smoothed by averaging them over a window of size 33 x 33. Then, an initial classification of the smoothed feature vectors takes place according to the testing phase of the architecture. The next step is to use a sliding window of size 18 x 18 to classify very small regions to the same class as the large region that surrounds them. This window classifies the feature vector that is mapped to the central pixel of the window, to the class that is dominant in this window. A class is dominant if the number of feature vectors that are classified as such are larger than the number of feature vectors that are classified as any other class. The next step is to consider the regions around the boundaries as an ambiguous class. It is possible that these regions are misclassified because the window that is used for smoothing averages features that correspond to different textures and exist on the different sides of the boundary. Smoothing windows whose central pixel is far away from the estimated boundaries more that half the width of the window, do not consist of more than one texture. The classification for feature vectors that are mapped to these pixels remains the same. The above algorithm is iteratively applied for every new estimated ambiguous classes for smoothing windows of sizes 25 x 25, 17 x 17, 9 x 9 and finally for the case that no smoothing windows are used. Using this iterative algorithm the boundaries between textures are not blurred significantly.

4. RESULTS

The method that is proposed in this paper is compared with the standard FAMNN with total number of nodes in the F_2^a field equal to 357 and 746. For the first FAMNN $\bar{\rho}_a$ was chosen to be 0, while for the second $\bar{\rho}_a$ was chosen to be 0.95. The proposed model is also compared to the 1-NN algorithm. The described iterative classification algorithm that was presented in 3.2.2 is applied in all classification methods. All four methods are compared with respect to classification of a testing set and the classification of the training set when it is subjected to uniform noise. The proposed method and 1-NN are compared with respect to the segmentation of images that consist of textures selected from the training set. The set that is referred as training set is the twenty textures from which the feature vectors that were used to train the networks are created.

The testing set consists of twenty textures. Each texture of the testing set is a different realization of the corresponding texture of the training set. The percentage of correct classification (PCC) for the testing set is given in Table 1. In this case all algorithms perform well with PCC close to 95%. The difference between the best and the worst PCC is 1.5 %. This difference is not significant especially because the selection of the testing set was based on visual estimation of similarity with the training set. In Figure 2 four of the textures of the training set and the corresponding ones from the testing set are presented.

The four algorithms are compared with respect to the classification performance in the case that the textures are subjected to additive uniform noise. The results are presented in Table 2 and in Figure 3. In the absence of noise and when the standard deviation of the noise is 7.2 the PCC is similar for all methods and it is close to 100%. The PCC is better for the proposed method if the standard deviation is 13, 18.8 and 24.5. The PCC of the proposed method is larger than the PCC of the 1-NN and the difference is almost constant and close to 4.5% for these cases. The difference between the PCC for the proposed method and the other two methods is even larger and it increases as the standard deviation of the noise increases.

The segmentation performance was also tested for the 1-NN and for the proposed method. The results are presented in Figures 4 and 5 and in Table 3. The purpose of this experiment is to show that the smoothing window and the window that is used for feature extraction are sufficiently large, so that more than one texture in the same image can be identified. The results were slightly better for the proposed method. It has been shown experimentally that iterative algorithms that use different sizes of smoothing window can help to avoid blurring at the regions close to the boundaries between textures. Here it is shown that there is about 2-3% misclassification at the boundaries because of the existence of more than one different texture in the same image.

5. CONCLUSIONS

In this paper a FAMNN variation is proposed. A modification of the input that defines the cluster, in which an input feature vector belongs, improves the PCC in the case where the textures are subjected to noise. This modification is feature dependent since it takes advantage of the characteristic that the FD feature values increase in the presence of noise. The PCC of the proposed variation is better than the original FAMNN and the 1-NN in a noisy environment. In the absence of noise the performance of the proposed variation is identical with the performance of the standard FAMNN. If the variance of the noise that contaminates the textures could be estimated, then the FAMNN could adapt so that the classification results could be further improved.

Also, the segmentation results show that the different parameters such as the size of the smoothing window and the size of the window where the feature vectors are computed are sufficient with the help of the iterative algorithm, so that different textures can be identified in the same image.

6. REFERENCES

1. Carpenter, G.A., Grossberg, S., Markuzon, N., Reynolds, J.H., and Rosen, D.B. , "Fuzzy ARTMAP: A neural network architecture for incremental supervised learning of analog multidimensional maps", *IEEE Transactions on Neural Networks,* No. 3, pp. 698-713, 1992.
2. B. Dubuc, C. Roques-Carmes, C. Tricot, and S. W. Zucker, "The variation method: a technique to estimate the fractal dimension of surfaces", *SPIE, Vol. 845, Visual Communications and image processing II,* pp. 241-248, 1987.
3. Anil K. Jain, and Farshid Farrokhinia, "Unsupervised texture segmentation using Gabor filters", *Pattern Recognition,* Vol. 24, No 12, pp. 1167-1185, 1991.
4. Dennis Dunn, and William E. Higgins, "Optimal Gabor filters for texture segmentation", *IEEE Transactions on image processing,* Vol. 4, No. 7, pp. 947-964, July 1995.
5. Andreas Teuner, Olaf Pichler and Bedrich J. Hosticka, "Unsupervised texture segmentation of images using tuned matched Gabor filters", *IEEE Transactions on image processing,* Vol. 4, No. 6, pp. 863-870, June 1995.
6. B.B Chaundhuri, and Nirupam Sarkar, "Texture segmentation using fractal dimension", *IEEE Transactions on pattern analysis and machine intelligence,* Vol. 17, No. 1, pp. 72-77, January 1995.
7. T. Kasparis, N.S. Tzannes, M. Bassiouni and Q. Chen, "Texture Description based on Fractal and Energy Features", *Computers and Electrical Engineering,* Vol. 21, No. 1, pp. 21-32, 1995

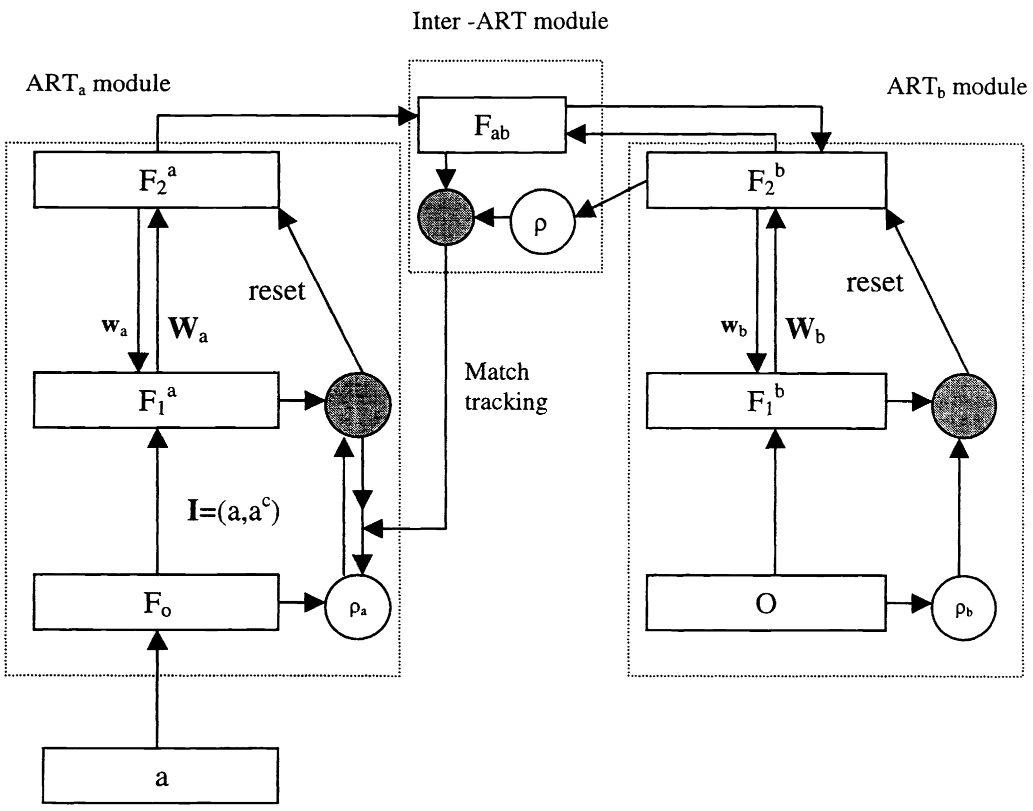

Figure 1: Block diagram of the ARTMAP architecture

TABLE 1: Percentage of correct classification % (PCC) for the testing set

Texture #	*Testing Set*			
	1-NN	*FAMNN* 367 nodes	*FAMNN* 746 nodes	*FAMNN (Suggested)* 746 nodes
1	97.5	97.6	97.3	99.0
2	99.9	96.7	98.9	99.5
3	99.1	99.4	99.2	100.0
4	100.0	100.0	100.0	100.0
5	50.2	57.0	58.7	58.2
6	99.9	99.7	99.7	100.0
7	96.9	87.7	93.1	89.2
8	98.7	99.5	99.3	98.2
9	99.3	97.9	99.8	99.4
10	100.0	100.0	99.7	100.0
11	100.0	93.6	99.1	100.0
12	100.0	100.0	98.6	98.4
13	99.8	99.4	100.0	99.0
14	88.8	83.1	85.9	70.9
15	100.0	100.0	100.0	100.0
16	100.0	100.0	100.0	99.3
17	96.9	99.8	99.2	90.9
18	99.7	99.5	100.0	99.6
19	98.0	92.8	96.2	91.8
20	100.0	99.1	100.0	100.0
ALL	*96.2*	*95.1*	*96.2*	*94.7*

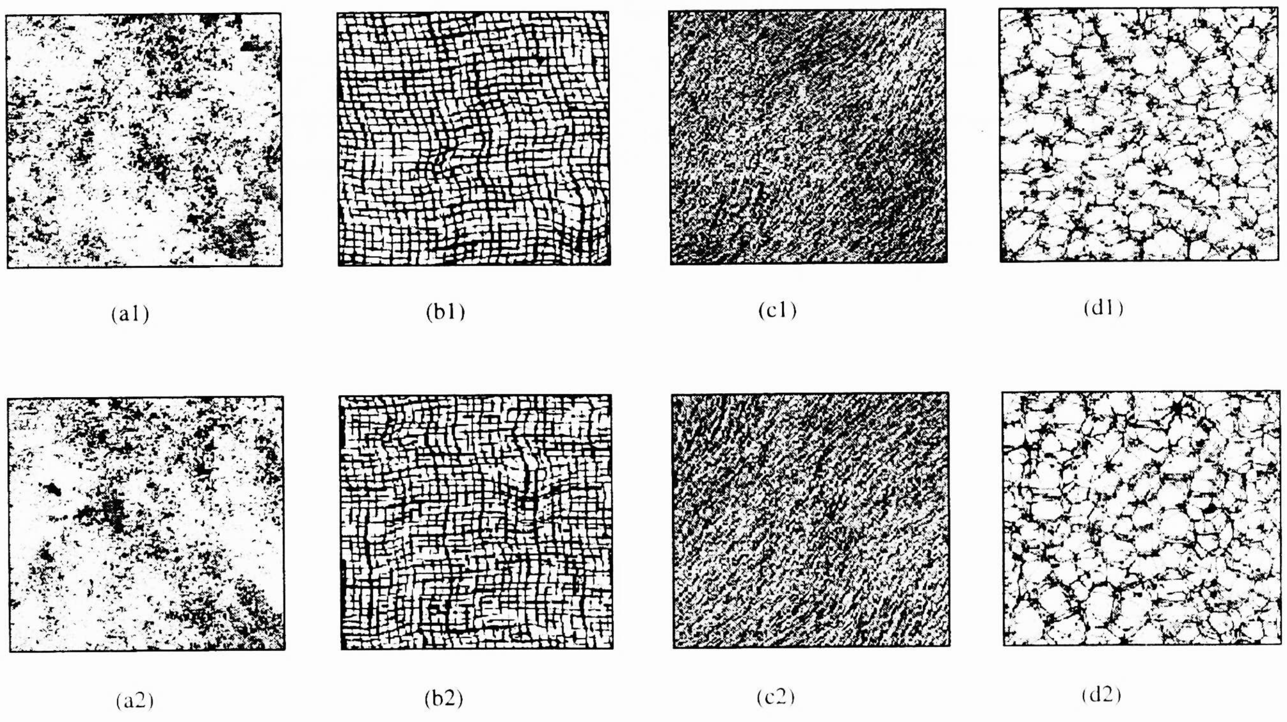

Figure 2: (a1),(b1),(c1),(d1) : Four textures from the training set
(a2),(b2),(c2),(d2) : The corresponding four textures from the testing set

TABLE 2: Percentage Of Correct Classification % (PCC) With Respect To St. Deviation Of Noise

Texture #	*No noise present*				*St. Deviation = 7.2*				*St. Deviation = 13*			
	1-NN	*FAMNN* *367 nodes*	*FAMNN* *746 nodes*	*FAMNN (Suggested) 746 nodes*	*1-NN*	*FAMNN* *367 nodes*	*FAMNN* *746 nodes*	*FAMNN (Suggested) 746 nodes*	*1-NN*	*FAMNN* *367 nodes*	*FAMNN* *746 nodes*	*FAMNN (Suggested) 746 nodes*
1	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	99.3	100.0	100.0
2	100.0	99.8	100.0	100.0	100.0	99.0	100.0	100.0	100.0	90.9	95.2	99.6
3	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
4	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
5	100.0	99.3	100.0	96.2	100.0	97.8	100.0	98.0	87.4	81.8	93.6	98.4
6	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
7	100.0	99.8	100.0	99.3	100.0	98.7	100.0	99.7	99.9	95.4	99.1	98.3
8	100.0	100.0	100.0	100.0	100.0	100.0	100.0	99.9	100.0	100.0	100.0	100.0
9	100.0	100.0	100.0	98.7	99.9	97.1	96.5	100.0	48.7	43.8	43.6	82.8
10	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
11	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	98.6	100.0	100.0
12	100.0	100.0	100.0	100.0	99.2	96.4	99.9	99.5	46.8	23.6	38.9	88.4
13	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
14	100.0	100.0	100.0	98.5	100.0	100.0	100.0	98.9	100.0	99.4	96.6	98.9
15	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
16	100.0	100.0	100.0	99.2	100.0	100.0	100.0	99.2	100.0	99.5	100.0	99.4
17	100.0	100.0	100.0	99.3	100.0	100.0	100.0	99.2	100.0	100.0	100.0	99.6
18	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	99.3	98.4	99.6	100.0
19	100.0	100.0	100.0	100.0	100.0	99.4	100.0	100.0	100.0	99.4	100.0	100.0
20	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
ALL	*100.0*	*99.9*	*100.0*	*99.6*	*100.0*	*99.4*	*99.8*	*99.7*	*94.1*	*91.5*	*93.3*	*98.3*

Texture #	*St. Deviation = 18.8*				*St. Deviation = 24.5*			
	1-NN	*FAMNN* *367 nodes*	*FAMNN* *746 nodes*	*FAMNN (Suggested) 746 nodes*	*1-NN*	*FAMNN* *367 nodes*	*FAMNN* *746 nodes*	*FAMNN (Suggested) 746 nodes*
1	100.0	98.6	97.4	100.0	96.9	89.6	90.1	100.0
2	76.8	45.0	37.9	91.0	10.0	3.0	0.0	15.9
3	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
4	100.0	99.1	99.7	100.0	83.6	78.1	68.8	82.3
5	85.8	67.7	71.3	88.9	67.1	45.5	22.7	68.9
6	99.4	92.0	97.9	100.0	89.1	10.5	35.0	98.0
7	98.2	94.7	97.5	98.3	85.7	55.8	86.1	93.7
8	100.0	98.2	100.0	99.6	97.9	88.5	86.1	99.5
9	2.6	1.1	3.3	18.3	0.0	0.0	0.0	0.0
10	93.3	70.3	29.3	99.5	1.6	0.0	0.0	12.7
11	95.5	64.0	93.7	99.6	67.8	1.5	37.9	82.8
12	0.0	0.0	0.0	1.3	0.0	0.0	0.0	0.0
13	100.0	100.0	100.0	100.0	100.0	94.5	98.7	100.0
14	96.4	94.4	90.0	94.7	81.7	78.5	80.3	83.5
15	100.0	94.9	98.5	100.0	96.5	68.8	81.7	99.7
16	99.7	93.8	99.4	99.5	81.1	63.6	85.1	96.8
17	100.0	100.0	100.0	99.7	100.0	98.4	100.0	99.8
18	57.0	48.1	52.3	93.1	2.3	1.7	1.4	15.9
19	100.0	98.7	99.7	100.0	93.0	88.9	96.3	98.4
20	99.4	97.9	99.0	99.4	93.1	83.5	88.5	93.5
ALL	*85.2*	*77.9*	*78.3*	*89.1*	*67.4*	*52.5*	*57.9*	*72.1*

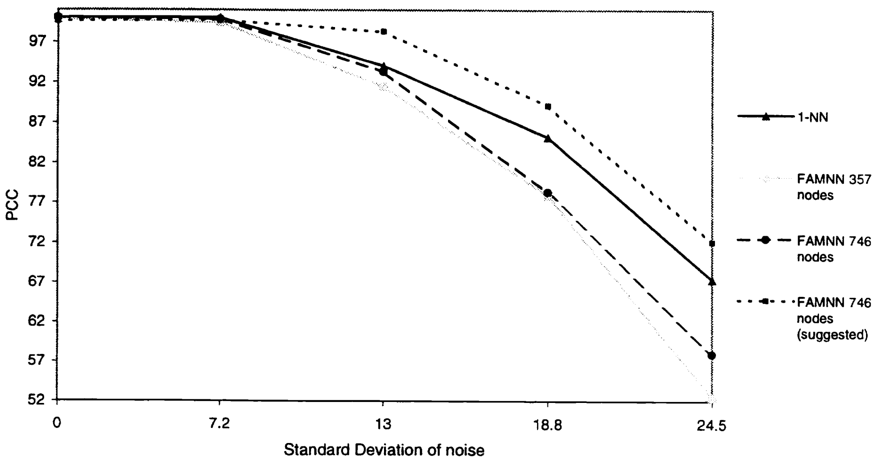

Figure 3: Comparison of the four classification approaches

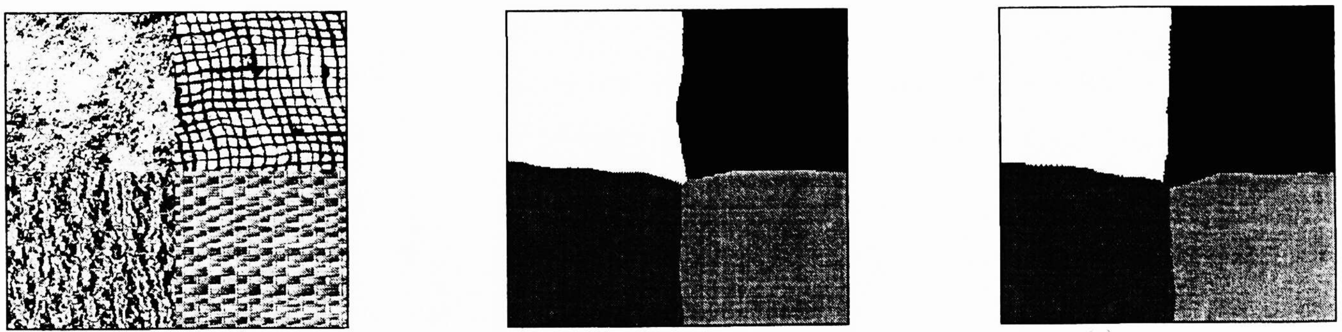
(a) (b) (c)

Figure 4 : Segmentation of an image that consists of four Textures from the Training Set. Textures 1, 3, 8, 11 (Up-left, Up-right, Down left, down-right respectively). (a) Original image, (b) Segmented image using Fractals and 1-NN, (c) Segmented image using Fractals and the suggested FAMNN

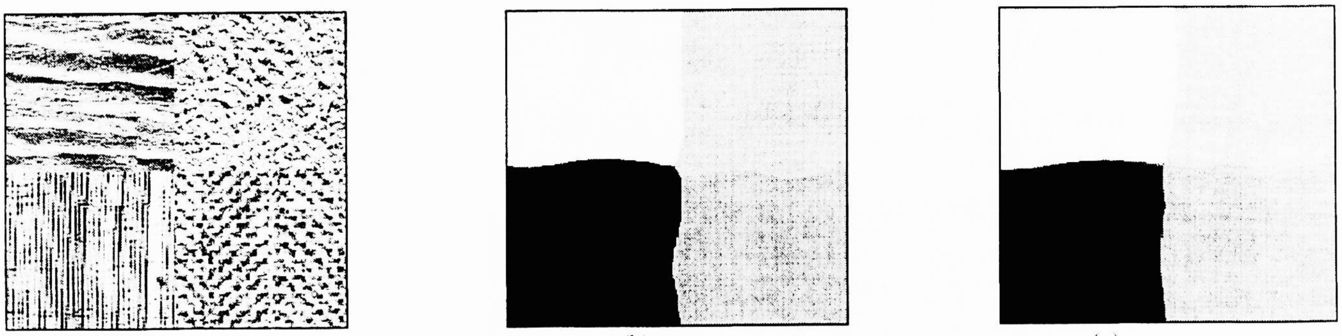
(a) (b) (c)

Figure 5 : Segmentation of an image that consists of four Textures from the Training Set. Textures 20, 16, 4, 15 (Up-left, Up-right, Down-left, down-right respectively). (a) Original image, (b) Segmented image using Fractals and 1-NN, (c) Segmented image using Fractals and the suggested FAMNN

TABLE 3 : Percentage of correct classification % (PCC) with respect to segmentation (a) for image in Figure 3, (b) for image in Figure 4

	Texture #				
	1	3	8	11	ALL
Suggested FAMNN	98.1	99.7	97.2	98.8	98.4
1-NN	98.8	97.8	98.6	95.3	97.6

(a)

	Texture #				
	20	16	4	15	ALL
Suggested FAMNN	98.3	96.3	95.5	98.8	97.2
1-NN	96.8	97.0	99.2	94.8	97.0

(b)