ARTIFICIAL NEURAL NETWORKS AND OTHER METHODS OF IMAGE CLASSIFICATION

M.SEETHA, * I.V.MURALIKRISHNA, MEMBER, IEEE ** B.L. DEEKSHATULU, LIFE FELLOW, IEEE, B.L.MALLESWARI, NAGARATNA, P.HEGDE

Associate Professor, HYDERABAD-500075,
*Director, R & D Division, JNTU, HYDERABAD-500072,
**Visiting Professor, HCU, HYDERABAD
E-mail: smaddala2000@yahoo.com, ivm@ieee.org

ABSTRACT

In digital image classification the conventional statistical approaches for image classification use only the gray values. Different advanced techniques in image classification like Artificial Neural Networks (ANN), Support Vector Machines (SVM), Fuzzy measures, Genetic Algorithms (GA), Fuzzy support Vector Machines (FSVM) and Genetic Algorithms with Neural Networks are being developed for image classification. Artificial neural networks can handle non-convex decisions. The use of textural features in ANN helps to resolve misclassification. SVM was found competitive with the best available machine learning algorithms in classifying high-dimensional data sets. Fuzzy measures show the detection of textures by analyzing the image by stochastic properties. The fundamental stochastic properties of the image are isolated by different kinds of stochastic methods, by non-linear filtering and by non-parametric methods. Fuzzy support vector machines (FSVM) was proposed to overcome the n-class problem in Support Vector Machines. In this using the decision functions obtained by training the SVM, for each class, a truncated polyhedral pyramidal membership function was defined. The genetic algorithm searches a space of image processing operations for a set that can produce suitable feature planes, and a more conventional classifier which uses those feature planes to output a final classification. The use of a hybrid genetic algorithm investigates the effectiveness of the genetic algorithm evolved neural network classifier and its application to the image classification of remotely sensed multispectral imagery. A comparative study of some of these techniques for image classification is made to identify relative merits.

Keywords: *Image classification, neural networks, support vector machines, fuzzy measures, genetic algorithms.*

1. INTRODUCTION

Digital image consists of discrete picture elements called pixels which are associated with a digital number represented as DN that depicts the average radiance of relatively small area with a scene. The range of DN values is normally 0 to 255. Digital image processing is a collection of techniques for the manipulation of digital images by computers. Classification generally comprises four steps: l. Pre-processing. e.g. atmospheric correction, noise suppression, and finding the band ratio, principal component analysis, etc, 2. Training: Selection of the particular feature which best describes the pattern, 3. Decision: Choice of suitable method for comparing the image patterns with the target patterns and 4: Assessing the accuracy of the classification. The informational data are classified into supervised and unsupervised systems.

2. TECHNIQUES OF IMAGE CLASSIFICATION

Image classification is an important task for many aspects of global change studies and environmental applications. Several classification algorithms have been developed from maximum likelihood classifier to neural network classifiers. This study emphasizes on the analysis and usage of different advanced classification techniques like Artificial Neural Networks, Support Vector Machines, Fuzzy Measures, Genetic algorithms and their combination for digital image classification. Finally the study depicts the comparative analysis of

different classification techniques with respect to several parameters.

A. Artificial Neural Network (ANN)

ANN is a parallel distributed processor [1] that has a natural tendency for storing experiential knowledge. They can provide suitable solutions for problems, which are generally characterized by non-linear ties, high dimensionality noisy, complex, imprecise, and imperfect or error prone sensor data, and lack of a clearly stated mathematical solution or algorithm. A key benefit of neural networks is that a model of the system can be built from the available data. Image classification using neural networks is done by texture feature extraction and then applying the back propagation algorithm.

1) Architecture of Neural Network and Texture Feature Extraction Algorithm:

Texture is characterized by the spatial distribution of gray levels in a neighborhood. In texture classification the aim is to assign an unknown sample image to one of set of known texture classes. Textural features are scalar numbers, discrete histograms or empirical distributions. In the design four textural features namely the angular second moment, contrast, correlation and variance are considered. Texture and tone have an inexpressible relationship to one another. They are always present in an image, although on occasion one property can overlook the other. In order to capture the spatial dependence of gray-level values, which contribute to the perception of texture, a two dimensional dependence, and texture analysis matrix is considered. Since, texture shows its characteristics by both pixel and pixel values, there are many approaches used for texture classification. Figure 1 shows the architecture of NN with combined gray value and textural features. In this, four layers consisting of three inputs, seven first layer hidden nodes, eleven second layer hidden nodes and five output nodes are considered.

The gray-tone co occurrence matrix is used for extracting textural features. It is a two dimensional matrix of joint probabilities Pd, r (i, j) between pairs of pixels, separated by a distance, d in a given direction r [2].

. For finding textural features for every pixel in the image every pixel is considered as a centre and followed by a 5 × 5 window about that centre pixel.
. The gray-tone matrix for that particular window is calculated and normalized.
. The gray level co-occurrence matrix namely, Ph, Pv, Prd and Pld for each pixel is then obtained. Here, Ph, Pv, Prd and Pld are respectively the 0^o , 90^o , 45^o and 135^o nearest neighbors to a particular resolution cell.
. Standard deviation and mean are now obtained for each of these matrices and the textural features are calculated.
. The particular entry in a normalized gray tone spatial dependence matrix is calculated for further reference, i.e., P(i,j), Px (i), Py (j), Px + y (k) and Px - y (k).
. Then textural features, the angular second moment, contrast, correlation and variance are calculated. After extracting the textural features the network is trained by standard back propagation algorithm (BKP).

2) Training the Combined Network using BKP

The following assumes the sigmoid function f (x)

$$f(x) = \frac{1}{1+e^{-x}} \quad \text{--- (1)}$$

The back propagation algorithm is implemented using following steps:

1. Initialize weights to small random values.
2. Feed input vectors X0, X1……., X6 through the network and compute the weighting sum coming into the unit and then apply the sigmoid function. Also, set all desired outputs d0,d1.. . . . . d5 typically to zero except for that corresponding to the class the input is from.
3. Calculate error term for each output unit as

$$\delta_j = y_j(1-y_j)(d_j - y_j) \quad \text{--- (2)}$$

Where dj is the desired output of node j; and y j is the actual output.

4. Calculate the error term of each of the hidden units as

$$\delta_j = x_j(1-x_j)\sum_k \delta_k w_{jk} \quad \text{--- (3)}$$

Where k is over all nodes in the layers above node j; and j is an internal hidden node.

5. Add the weight deltas to each of

$$W_y(t+1) = W_y(t) + \eta \delta_j x_i$$

--- (4)

All the steps excepting 1 are repeated till the error is within reasonable limits and then the adjusted weights are stored for reference to the recognition algorithm.

B. Support Vector Machines

The support vector machine (SVM) is superior of all machine learning algorithms. SVM employs optimization algorithms to locate the optimal boundaries between classes. The optimal boundaries should be generalized to unseen

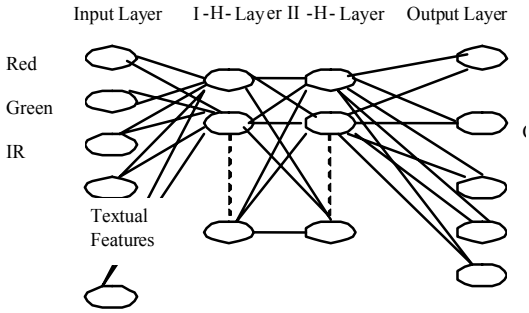


Figure:1: Architecture of NN with combined gray v
textural features

samples with least errors among all possible boundaries separating the classes, therefore minimizing the confusing between classes. The applicability of SVM for image classification is explored in this study.

1) Development of SVM

The support vector machine (SVM) is a machine learning algorithm based on statistical learning theory. There are a number of publications detailing the mathematical formulation and algorithm development of the SVM [8, 9, and 10]. The inductive principle behind SVM is structural risk minimization (SRM). The risk of a learning machine (*R*) is bounded by the sum of the empirical risk estimated from training samples (*R*emp) and a confidence interval (ψ): $R \leq R\text{emp} + \psi$ [8]. The strategy of SRM is to keep the empirical risk (*R*emp) fixed and to minimize the confidence interval (ψ), or to maximize the margin between a separating hyper plane and closest data points (Figure 2). A separating hyperplane refers to a plane in a multi-dimensional space that separates the data samples of two classes. The optimal separating hyperplane is the separating hyperplane that maximizes the margin from closest data points to the plane. Currently, one SVM classifier is able to separate only two classes. Integration strategies are needed to extend this method to classifying multiple classes.

2) The optimal separating hyperplane

Let the training data of two separable classes with k samples be represented by(x_1, y_1), (x_k, y_k) where $x \epsilon R^n$ is an n-dimensional space, and $y \epsilon \{+1, -1\}$ is class label. Suppose the two classes can be separated by two hyperplanes parallel to the optimal hyperplane (Figure 2(*a*)):

$$w.x_i + b \geq 1$$

--- (5)

$$w.x_i + b \leq -1$$

--- (6)

where **w**= $(w_1 \ldots w_n)$ is a vector of n elements. Inequalities (5) and (6) can be combined into a single inequality:

$$y_i[w'x_i + b] \geq 1$$

--- (7)

As shown in Figure 2, the optimal separating hyperplane is the one that separates the data with maximum margin. This hyperplane can be found by minimizing the norm of w, or the following function:

$$F(w) = {}^{1}\!/_{2}(w'w)$$

--- (8)

The saddle points of the following Lagrange gives solutions to the above optimization problem:

$$L(w,b,\alpha) = {}^{1}\!/_{2}(w'w) - \sum_{i=1}^{k} \alpha_\iota y_i[w'x_i + b] - 1\}$$

--- (9)

where $\alpha_i \geq 0$ are Lagrange multipliers [11]. The solution to this optimization problem requires that the gradient of L (w, b, α) with respect to w and b vanishes, giving the following conditions:

$$w = \sum_{i=1}^{k} y_i \alpha_\iota x_i$$

--- (10)

$$\sum_{i=1}^{k} \alpha_\iota y_i = 0$$

--- (11)

By substituting (10) and (11) into (9), the optimization problem becomes: maximize

$$L(\alpha) = \sum_{i=1}^{k} \alpha_i - \frac{1}{2} \sum_{i=1}^{k} \sum_{j=1}^{k} \alpha_i \alpha_j y_i y_j (x_i ' x_j)$$

--- (12) under constraints, $\alpha_i \geq 0$ i=1,.. , *k*.

Given an optimal solution $\alpha^0 = (\alpha^0_{1,}....,\alpha^0_k)$ to (12), the solution w^0 to (9) is a linear combination

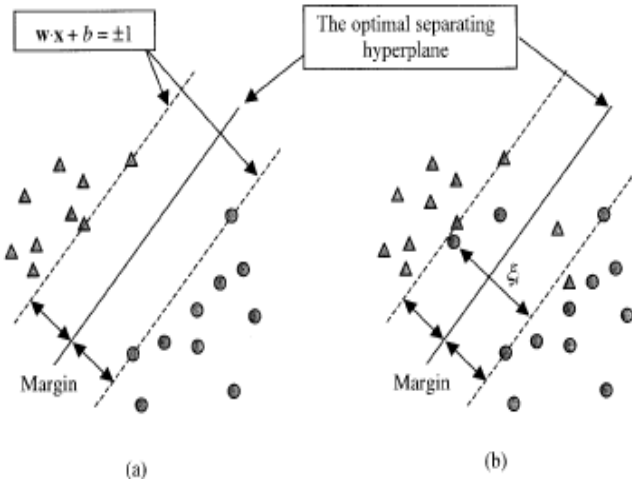


Figure 2: The optimal separating hyperplane between (*a*) separable samples and (*b*) non-separable data samples.

of training samples:

$$w^0 = \sum_{i=1}^{k} y_i \alpha_i^0 x_i$$

--- (13)

According to the Kuhn–Tucker theory [11] only points that satisfy the equalities in (5) and (6) can have non-zero coefficients α_i^0 .These points lie on the two parallel hyperplanes and are called support vectors (Figure 2). Let $\mathbf{x}^0(1)$ be a support vector of one class and $\mathbf{x}^0(-1)$ of the other, then the constant b^0 is calculated as follows:

$$b^0 = \tfrac{1}{2}[w^0 ' x^0(1) + w^0 ' x^0(-1)]$$

--- (14)

The decision rule that separates the two classes can be written as:

$$f(x) = sign \left(\sum_{\sup portvector} y_i \alpha_i^0 (x_i ' x) - b^0 \right)$$

--- (15)

3) Handling non-separable cases

An important assumption to the above solution is that the data are separable in the feature space. It is easy to check that there is no optimal solution if the data cannot be separated without error. To resolve this problem, a penalty value *C* for misclassification errors and positive slack variables ξ_i are introduced (Figure 2(*b*)).

These variables are incorporated into constraint (5) and (6) as follows:

$$w' x_i + b \geq 1 - \xi_i \quad \text{for} \quad y_i = 1$$

--- (16)

$$w' x_i + b \leq -1 + \xi_i \quad \text{for} \quad y_i = 1$$

--- (17)

$$\xi_i \geq 0 \qquad i = 1,\ldots\ldots,k$$

--- (18)

The objective function (8) then becomes

$$F(w, \xi) = \tfrac{1}{2}(w'w) + C \left(\sum_{i=1}^{k} \xi_i \right)^l$$

--- (19)

C is a preset penalty value for misclassification errors. If *l*=1, the solution to this optimization problem is similar to that of the separable case.

4) Implementation of Support vector machines

To generalize the above method to non-linear decision functions, the support vector machine implements the following method: it maps the input vector *x* into a high-dimensional feature space *H* and constructs the optimal separating hyperplane in that space. Suppose the data are mapped into a high-dimensional space *H* through mapping function Φ:

$$\Phi : R^n \rightarrow H$$

--- (20)

A vector *x* in the feature space can be represented as Φ(**x**) in the high-dimensional space *H*. Since the only way in which the data appear in the training problem (8) are in the form of dot products of two vectors, the training algorithm in the high dimensional space *H* would only depend on data in this space through a dot product, i.e. on functions of the form $\Phi(\mathbf{x}_i)' \Phi(\mathbf{x}_j)$. Now, if there is a kernel function *K* such that

$$K(x_i, x_j) = \Phi(x_i) ' \Phi(x_j)$$

--- (21)

then K is used in the training program without knowing the explicit form of Φ. Thus if a kernel function *K* can be found, a classifier can be used in the high-dimensional space without knowing the explicit form of the mapping function for training.. The optimization problem (12) is rewritten as:

$$L(\alpha) = \sum_{i=1}^{k} \alpha_i - \frac{1}{2} \sum_{i=1}^{k} \sum_{j=1}^{k} \alpha_i \alpha_j y_i y_j K(x_i, x_j)$$

(22)

and the decision rule expressed in equation (11) becomes:

$$f(x) = sign\left(\sum_{\sup port\ vector} y_i \alpha_i^0 K(x_i, x) - b^0 \right)$$

(23)

By the systematic development of SVM, the kernel function plays a major role in locating complex decision boundaries between classes. By having input data into dimensional space, the kernel function converts non-linear boundaries in the original data space into linear ones in the high dimensional space. Hence the selection of kernel function and appropriate values for corresponding kernel parameters, affect the performance of the SVM.

C. Fuzzy Measures

There are several image processing algorithms for the detection of information in the form of sound defined features such as edges, skeletons, and contours etc as in [13, 14, and 15]. In Fuzzy measures, different stochastic relationships are identified to describe properties of an image. The different types of stochastic are collected is a set of properties, where the members of this set are fuzzy in their contribution. The fuzzy measure gives the possibility to describe different types of stochastic properties in the same form. If the fuzzy property is more related to a region, then a fuzzy measure is used. Fuzzy function is used if a stochastic property is to be described by a particular distribution of gray values. The fusion of these two stochastic properties is represented as a fuzzy measure and fuzzy function defines on an area which is achieved by a fuzzy integral. The result of fuzzy integral is a new fuzzy measure.

1) Stochastic Properties of the Image

For the extraction of stochastic properties different methods are used, because the stochastic properties are a composite and decomposition is only possible, if the composed properties are calculated by different aspects. The different composed properties will be mapped on different spaces and isolated. The filtering is adapted for stochastic changing of the grey values related to the neighbor pixels. The regions with different stochastic variations are well selected by application of wavelet transforms with selected constants. The dynamic of the stochastic between the pixel-points describes many relevant properties of the textures. Consequently, the stochastic properties are modulated by a system of coupled stochastic differential equations of the form:

$$dy_1(x) = F_1(y_1, y_2, ..., y_n)dx + G_1 n(x)$$
$$dy_2(x) = F_2(y_1, y_2, ..., y_n)dx + G_2 n(x)$$
$$.....$$
$$dy_m(x) = F_m(y_1, y_2, ..., y_n)dx + G_m n(x)$$

Here, y_1, y_2 ,….,y_n the m components of the stochastic and F are the m non-linear matrices for the relationships between the components. The m coefficients G are the gain factors for the noise n and x represents the pixel-point or the region in the image. From these equations the expectation values for y are be obtained by Martingale technique [16]. The difference of the measure values and the calculated expectation values represents the stochastic structure of the region within the image. This set of stochastic characteristics is a contribution to the basis of properties for the separation of textures. These obtained stochastic properties have to be represented in a special manner for a better estimation by the fuzzy measure and the fuzzy function and is applied in [17, 18].

2) Stochastic Information by Fuzzy Measures

The mathematical basis for including the importance of a stochastic property is the fuzzy measure. By the fuzzy measure the properties described by different kinds of relationships are mapped into the closed interval [0, 1]. The fuzzy measure, as defined in [19], has a term with the combination of all elementary fuzzy measures multiplied by a factor λ. The factor λ has an effect similar to a weight factor for the interaction between the properties. If λ= 0 then g can be used as a probability measure.

The coupling of the elementary fuzzy measures (densities) $g_1(x_1)$ over the elementary region x_1 with another elementary fuzzy measure $g(x_2)$ over the other elementary region x_2 is defined by:

$$g_\lambda(x_1 \cup x_2) = g_\lambda(x_1) + g_\lambda(x_2) + \lambda g_\lambda(x_1) g_\lambda(x_2)$$

-- (24)

where $\lambda = (1+ \lambda g(x_1))\ (1+ \lambda\ g(x_2) + \lambda\ g_\lambda(x_1)\ g_\lambda(x_2))^{-1}$ is a coupling constant used as a substitution for the loss of additivity. For a set of elements A= $\{x_i\}$ the relationship above is used recursively to give:

$$g(A) = \sum_{i=1}^{n} g(x_i) + \lambda \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} g(x_i) g(x_j) + .. + \lambda^{n-1} g(x_1)..g(x_n)$$

--- (25)

This is written as product

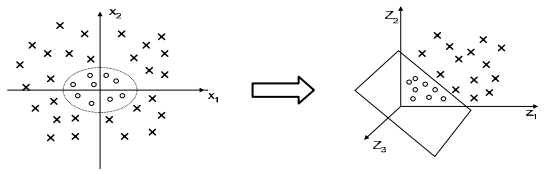


Fig. 3 mapping nonlinear data to a higher dimensional feature space

$$g(X)=\frac{1}{\lambda}\left[\prod_{x_j\in X}(1+\lambda g(x_i)-1\right] \quad \text{where} \quad \lambda\neq 0 \quad \text{--- (26)}$$

The coupling parameter λ is obtained by solving the equation

$$1+\lambda=\left[\prod_{x_j\in X}(1+\lambda g(x_i))\right]$$

--- (27)

The mathematical concept for calculating the measure for the coupled elementary properties for small areas is shown here.

3) Stochastic Information By Fuzzy Functions

The fuzzy functions are mostly values over single pixel points. The values of the neighbor pixel are of stochastic nature and normally not directly correlated with this value. These fuzzy functions are described normally by a characterization over a threshold. Outside of such a characteristic threshold the values depend very weakly on the real value. Inside the interval the values generate fuzzy properties for the adapted condition. These fuzzy functions are also normalized and mapped on an interval given by a boundary. Whereas the fuzzy measure is better adapted for effects represented in special regions, the fuzzy function characterizes the stochastic change over a region of a fuzzy measure. The values are combined in the similar manner as with the fuzzy measure.

4) Fusion of Fuzzy Properties by Fuzzy Integrals

The values over the possible region of textures represented by a fuzzy measure $g(x_{k,\,l})$ are connected with the values h $(x_{i,j})$ at the pixels representing the strength of the properties for a texture. The value h $(x_{i,j})$ is described by a fuzzy function where the values are normalized to 1. The functional relationship between the fuzzy measure and the fuzzy function is represented by the fuzzy integral. For the fuzzy integral the definition of a fuzzy integral of [19] is used, because it is well adapted to the problem of detection of textures. By [19] the fuzzy measure is combined with the fuzzy function in the form

$$f_A h_\alpha(x)\oplus$$

$$dg=\sup_{\alpha\in[0,1]}\{\min[\alpha,g(A\cap H_{\alpha)}]\},$$

$$H_\alpha=\{x|h_\alpha\geq\alpha\}$$

--- (28)

Here h_α is the cut of h at the constant α. For h_α (x), the values at the pixel-points are used, representing the property of a texture. α is the threshold where the assumption is fulfilled, that the property is used in the minimal condition. The region A is given as the image region where surely a specific texture is expected. It may be also the whole image for the pixel region and the whole possible range for a fuzzy function. The important property of a fuzzy measure is that its value is mapped on the closed interval [0, 1]. This is given by the calculated value of the fuzzy integral. This gives the possibility to use the result of a fuzzy integral as a new fuzzy measure g_2

$$g_{2_{i,j}}(A)=f_A h_\alpha(x)\oplus dg_1$$

--- (29)

This newly produced fuzzy measure is linked with the region obtained by another stochastic property for the texture so that f_2 obtained

$$f_2=f_A h'_{\alpha'}(x)\oplus dg_2$$

--- (30)

In the next step the next region of another property of the contrail is combined with this fuzzy function f_2. In such away a set of fuzzy functions {f1, f_2,…f_n} is obtained by fuzzy measures {g_1,g_2,…g_3}. The summation of all combinations of fuzzy measures with fuzzy functions makes sure, that all possible properties in all combinations, which should be considered, are used. In such a way an image is obtained, where the (grey) values represent a measure for the membership to the texture. In this way different elementary stochastic properties are combined in many ways for the extraction of relevant information. In order to achieve this different approaches have to be applied for the elimination of the elementary stochastic properties within an image.

D. Fuzzy Support Vector Machines

In support vector machines, the original input space is mapped into a high dimensional dot product space called feature space and in the

feature space the optimal hyper plane is determined to maximize the generalization ability [8, 9]. To overcome the problem of misclassification regions in multiclass support vector machines, fuzzy support vector machines are proposed in this study. In training the support vector machines, an n-class problem is converted into *n* two-class problems. For each two-class problem, the decision function that maximizes the generalization ability is determined. For a two-class problem, the m-dimensional input **x** is mapped into the 1-dimensional ($l \geq m$) feature space **z.** Then in the feature space **z** the quadratic optimization problem is solved to separate two classes by the optimal separating hyperplane.

1) Multiclass Support Vector Machines

In conventional support vector machines, an n class problem is converted into n two-class problems and for the ith two-class problem, class *i* is separated from the remaining classes. Let the ith decision function that classifies class *i* and the remaining classes be

$$D_i(x) = w_i^t x + b_i$$

--- (31)

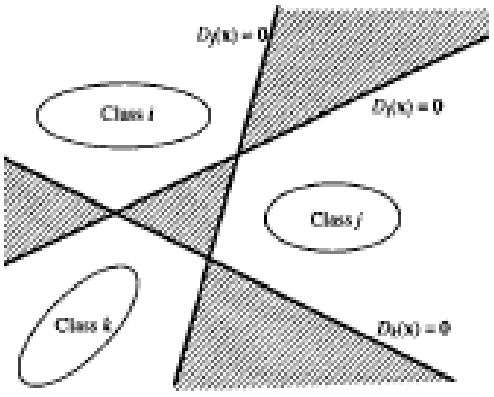
The hyperplane $D_i(x) = 0$ forms the optimal separating hyperplane and the support vectors belonging to class *i* satisfy $D_i(x) = 1$ and to those belonging to the remaining class satisfy $D_i(x) = -1$**.** For conventional support vector machine, if for the input vector x

$$D_i(x) > 0$$

--- (32)

is satisfied for one *i***,** x is classified into class *i.*

But if (32**)** is satisfied for plural *i's,* or there is no *i* that satisfies (32), x is unclassifiable (see Fig. 3).



nclassifiable region by the two-class formulation

To solve this problem, pairwise classification [20] is proposed.

In this method, the n-class problem is converted into n(n - 2)/2 two-class problems, which cover all pair of classes. Let the decision function for class *i* against class *j,* with the maximum margin, be

$$D_{ij}(x) = -D_{ji}(x)$$

--- (33)

where $D_{ij}(x) = -D_{ji}(x)$**.** For the input vector x calculate

$$D_i(x) = \sum_{i=1,\ldots n}^{n} sign(D_{ij}(x))$$

--- (34)

and classify x into the class

$$\arg\max_{i=1,\ldots,n} D_i(x)$$

--- (35)

In this formulation, however, unclassifiable regions

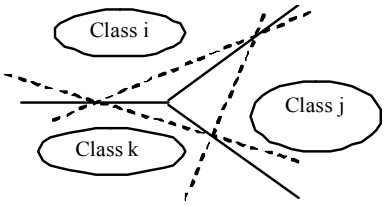


Figure 5: Class Boundary with membership function

remain, where some $D(x_i)$ have the same values. In order to resolve unclassifiable regions, fuzzy support vector machines are used.

2) Fuzzy Support Vector Machines

In fuzzy support vector machines, fuzzy membership functions are given for the same classification results for the data that satisfy (32). To do this, for class *i* one-dimensional membership functions $m_{ij}(x)$ are defined on the directions orthogonal to the optimal separating hyperplanes Dj (x) = 0 as follows:

1. For *i=j*

$$m_{ii}(x) = \begin{cases} 1 & for \quad D_i(x) > 1, \\ D_i(x) & otherwise. \end{cases}$$

--- (36)

2. For i≠j

$$m_{ij}(x) = \begin{cases} 1 & for \quad D_j(x) > -1, \\ -D_i(x) & otherwise. \end{cases}$$

--- (37)

Since only the class *i* training data exist when $D_i \geq 1$, that the degree of class *i* is assumed to be 1, and otherwise, $D_i(x)$. Here the negative degree of

TABLE I
THE GENES USED IN GENETIC PROGRAMMING SYSTEM AND THEIR DESCRIPTION

Gene Abbre-viation	Image Processing Operation	Inputs/ Outputs/ Params	Notes
ADDP	Add planes	2/1/0	Basic mathematical operations. ADDS adds a scalar, which may be negative, to its input. DIFF is like SUBP but outputs the absolute values. LINCOMB outputs a linear combination of its two inputs, in proportion specified by its one parameter, which takes a value between 0 and 1.
ADDS	Add scalar	1/1/1	
SUBP	Subtract planes	2/1/0	
DIFF	Absolute diff.	2/1/0	
NDI	Normalized diff.	2/1/0	
MULTS	Multiply by scalar	1/1/1	
NEG	Negate plane	1/1/0	
MULTP	Multiply planes	2/1/0	
SQRT	Square root	1/1/0	
LINCOMB	Linear comb.	2/1/1	
MIN	Minimum	2/1/0	Logical operations.
MAX	Maximum	2/1/0	
CLIP HI	Clip high	1/1/1	Thresholding operations. CLIP HI truncates any pixel values above a value set by its param.
CLIP LO	Clip low	1/1/1	
SAVAR	Spectral angle variance	2-16/1/2	Spectral angle operations. SAVAR and SADIST look at two circular neighborhood regions around each pixel, of size defined by their two params. SADIF returns the difference between the mean spectral angle of both regions.
SADIF	Spectral angle difference	2-16/1/2	
LAWB	Lawe's texture measure	1/1/0	Neighborhood operations. All these operations take a single plane as input and produce a single output plane. LAWB, LAWD, are widely-used texture measures, developed by Lawe, that return zero if the neighborhood contains all the same value of pixel, and some other value otherwise, depending upon the distribution of pixel values. Most of the other operators are familiar image processing or morphological operators, whose description can be found in any good book on image processing.
LAWD	Lawe's texture measure	1/1/0 1/1/1	
LAPLAC5	5x5 Laplacian	1/1/0	
SD	Standard deviation	1/1/1	
EROD	Erode	1/1/1	
DIL	Dilate	1/1/1	
OPEN	Open	1/1/1	
CLOS	Close	1/1/1	
OPCL	Open-close	1/1/1	

membership is allowed. For $i \neq j$, class i is on the negative side of $D_j(x) = 0$. In this case, support vectors may not include class i data but when $D_i(x) \leq -1$, we assume that the degree of membership of class i is 1, and otherwise, $-D_j(x)$.

The class i membership function of x is defined using the minimum operator for $m_{ij}(x)$ ($j = 1, \ldots, n$):

$$m_i(x) = \min_{j=1,\ldots,n} m_{ij}(x)$$

--- (38)

In this formulation, the shape of the membership function is a polyhedral pyramid (see Figure 4).

Now the datum **x** is classified into the class

$$\arg\max_{i=1,\ldots,n} D_i(x)$$

--- (39)

If x satisfies

$$D_k(x) \begin{cases} > 0 & for \quad k=i \\ \leq 0 & for \quad k \neq i, k=1,\ldots,n \end{cases}$$

--- (40)

from (36) and (37), $m_i(x) > 0$ and $m_j(x) \leq 0$ ($j \neq i$, $j = 1, \ldots, n$) hold. Thus, x is classified into class i. This is equivalent to the condition that the condition that (32) is satisfied for only one i. Now suppose (32) is satisfied for $i_1, \ldots, i_l$ ($l > 1$). Then, from (36) to (38), $m_k(x)$ is given as follows.

1. $k \in i_{1,\ldots,} i_l$

$$m_k(x) = \min_{j=i_1,\ldots,i_l, j\neq k} -D_j(x)$$

--- (41)

2. $k \neq j (j = i_1 \ldots, i_l)$

$$m_k(x) = \min_{j=i_1,\ldots,i_l} -D_j(x)$$

--- (42)

Thus the maximum degree of membership is achieved among $m_k(x)$, $k = i_1, \ldots, i_l$. Namely, $D_k(x)$ is maximized in $k \in (i_1, \ldots, i_l)$.

Let (32) be not satisfied for any class. Then

$$D_i(x) < 0 \qquad \text{for} \qquad j=1,\ldots,n.$$

--- (43)

Then (38) is given by

$$m_i(x) = D_i(x)$$

--- (44)

According to above formulation, the unclassified regions shown in Figure 3 are resolved as shown in Figure 5 and generalization ability of FSVMs is the

same with or better than that of the conventional SVMs. It was found that fuzzy support vector machines for classification is superior to conventional support vector machines.

E. Genetic Algorithms

The techniques of image classification ranging from maximum likelihood to neural networks depend on feature vectors formed by the intensity values in each spectral channel for each pixel. But the spectral information alone is not sufficient to exactly identify a pixel. The features of its neighborhood, like texture, or the average value of nearby pixels are necessary to get good spectral information. The different kinds of spatial content information could also be added into the pixel feature vector as additional feature dimensions. So there are a large number of choices for additional feature vectors that could make classification better

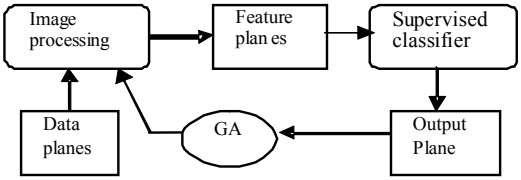


Fig ure 6:T he Structure of Genetic programming System

than just having the raw spectral values as feature vectors. Hence to choose these features automatically a new evolutionary hybrid genetic algorithm is used.

1) The Genetic Programming System

The genetic programming system based on a linear chromosome [22] manipulates image processing programs that take the raw pixel data planes and transform them into a set of feature

ADDS,Rd1, Ws1,0.2	NDI,rD3, rS1,wS1	OPCL, rS1,wS2	SQRT, rS1,wS1	CLIP_HI, rS2,wS2,0.1

Scratch1 <= ADDS(Data1, 0.2) Scratch1 <= NDI(Data3, Scratch1)
Scratch2 <= OPCL(Scratch1) Scratch1 <= SQRT(Scratch1)
Scratch2 <= CLIP_HI(Scratch2, 0.1)

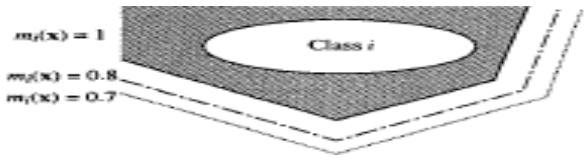


Figure 4: Contour lines of the class i membership function

planes. This set of feature planes is in effect just a multi-spectral image of the same width and height as the input image, but perhaps having a different number of planes, and derived from the original

image via a certain sequence of image processing operations. The system then applies a conventional

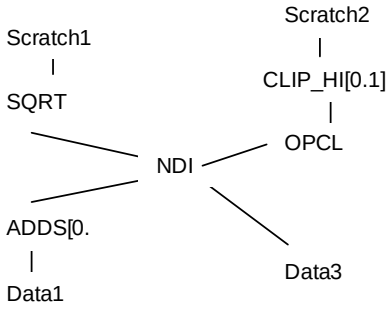


Figure 7: Three Equivalent views of a chromosome First and second shows the raw linear genome representation and Lines of code version of the genome. Third one shows the graph representation of the same genome.

supervised classification algorithm to the feature planes to produce a final output image plane, which specifies for each pixel in the image, whether that feature is there or not. Figure 6 illustrates this hybrid scheme. In this structure finally raw data planes are transformed into a set of feature planes by an image processing program that is evolved by genetic algorithm.

In the system a fixed-length linear chromosome and standard one-point crossover are used. According to [23], these two are significantly better than more flexible crossover schemes. A single chromosome is made up of a string of genes, each one of which corresponds to a particular image processing operation. Each gene has one or more inputs, and one or more outputs. An input can be taken from any one data plane in the original image (there are as many data planes as there are spectral channels), or from any one 'scratch plane'. Scratch planes are temporary holding places where a single image plane can be held. The gene performs some image processing operation on its inputs and produces one or more planes of output data. Each of these planes is written to a different scratch plane. The whole chromosome is evaluated by starting with the gene at the left end, and sequentially stepping through the genes in order, one-by-one. It is a requirement when chromosomes are created that no gene is allowed to read from a scratch plane that has not been written to by at least one gene to the left of it. In addition, the requirement imposed is that all scratch planes must be written to at least once during a chromosome's execution. The feature planes which are passed on to the supervised classifier are specified by the user as a subset of the scratch and data planes. Figure 7 shows the genome representation translation into an image processing pipeline. It is quite possible that a gene will write its output to a scratch plane, which is then overwritten by another gene before it is ever used.

In this case that gene is irrelevant, as they are any genes that write data to scratch planes that are only read by irrelevant genes before being overwritten. Hence, although the chromosome length is fixed, the effective program length can vary significantly. In this system, an efficient graph analysis of the chromosome is performed and irrelevant genes are determined. Those genes are kept in the chromosome, but for efficiency, are never actually executed. Table I lists some of the genes used and their description.

Each gene corresponds to a different image processing operation, but the details of that operation can be influenced by gene parameters. Different genes have different numbers of parameters, and each parameter is associated with a fixed set of attributes that determine such things as what range it is randomly initialized within when that gene is first created, what range of values it can possibly take, and how it is affected by mutation. There are three kinds of parameters:1) float parameters are initialized to a random floating point number in the initialization range, and are mutated by a floating point offset that is Gaussian distributed with a standard deviation given by that parameter's delta attribute; 2) integer parameters are initialized to a random integer in the initialization range, and are mutated by an integer offset that is uniformly distributed in a range given by plus or minus the delta attribute; 3) symbolic parameters are like integers, but when mutated are simply re-initialized randomly. In general, genes produce output that is roughly on the order of the same scale as their input. Thus by using GA a robust classifier is developed.

F. Genetic Algorithms and the Neural Network

The most widely used neural network is the multilayer perception (MLP), in which the connection weight training is done by a back propagation learning algorithm [25]. Despite its popularity as an optimization tool for neural network training, this gradient descent technique has several problems like premature convergence and efficiency of differential operation. Genetic algorithms [26] have an efficient search method for a complex problem space and are used as powerful optimization tools. With genetic algorithm the neural network connection weights are determined by the architecture and the learning task.

1) Combination of Neural Network and Genetic Algorithm

A three-layer feed-forward neural network with m inputs (channels) and k outputs (categories), and l hidden nodes is assumed in this study. Each neuron in the hidden layer uses *sigmoid* function $f(x)$ as its threshold function, and each neuron in the output layers uses *Purelin* function $p(x)$ as its threshold function. The neuron output of hidden node h ($1 \le h \le l$) and output node q ($1 \le q \le k$) can be expressed as:

$$z_h = f(W^T X) = f\left(\sum_{i=1}^{m} w_i x_i - \delta_n\right)$$

--- (45)

$$o_q = p\,(V^T Z) = p\left(\sum_{i=1}^{l} v_i z_i - \delta_q\right)$$

--- (46)

respectively, where the superscript T stands for a vector transpose, $W = [\,\omega_1, \omega_2, \ldots, \omega_i, \ldots, \omega_m]$ is the weight connection vector between the input nodes and hidden node h, $V = [v_1, v_2, \ldots, v_i, \ldots, v_k]$ is the weight connection vector between the hidden nodes and output node q, $X = [x_1, x_2, \ldots, x_i, \ldots, x_m]$ is the input vector for each hidden node, and $Z = [z_1, z_2, \ldots, z_i, \ldots, z_m]$ is the output vector of the hidden nodes. δ_h and δ_q are the corresponding biases for hidden node h and output node q. z_h and o_q are the output neuron responses for node h and node q, respectively. The Sigmoid function $f(x)$ is defined as:

$$f(x) = \frac{1}{1+e^{-x}}$$

--- (47)

where $x \in [-\infty, +\infty]$.And the Purelin function $p(x)$ is defined as:

$$p(x) = \alpha x + \beta$$

--- (48)

where α is a non-zero constant, β is the bias; and $\alpha, \beta \in [-\infty, +\infty]$.

Assuming a set of pattern samples $X=\{X_1, X_2, \ldots, X_n\}$, where n is the number of samples, and each sample Xi in set X is a m-dimensional feature vector; let $T=\{T_1, T_2, \ldots, T_i, \ldots, T_n\}$ as set X's corresponding output classes, $Ti = [t_1, t_2, \ldots, t_j, \ldots, t_k]$ is a k-dimensional class vector. If the target class for a specific sample is j ($1 \le j \le k$), then we have $t_j = 1$, otherwise $tj = 0$. For simplicity, denote o_{ij} as the j^{th} actual neuron output for the input training sample X_i at the output layer, while t_{ij} as its desired response. The mean square error function for this neural network could be described as:

$$\varepsilon(net) = \frac{1}{n.k}\sum_{i=1}^{n}\sum_{j=1}^{k}(t_{ij} - o_{ij})^2$$

--- (49)

where ε is mean square error, *net* is the neural network.

The combination of genetic algorithm and neural network for weight training consists of three major phases. The first phase is to decide the representation of connection weights, i.e., whether we use a binary strings form or directly use a real number form are to represent the connection weights. The second step is the evaluation on the fitness of these connection weights by constructing the corresponding neural network through decoding each genome and computing its fitness function and mean square error function. The third one is applying the evolutionary process such as selection, crossover, and mutation operations by a genetic algorithm according to its fitness. The evolution stops when the fitness is greater than a predefined value or the population has converged.

The technical design of the evolutionary strategy of connection weights training can be described as:

1) Decode each individual (genotype) in the current generation into a set of connection weights. The choice of real coded genotype representation can search the potential solutions more precisely in feature space than binary representation.

2) Evaluate each set of the connection weights by constructing the corresponding neural network structure and computing its total mean square error between actual and target outputs. The higher the error, the lower the fitness. The fitness function δ is defined as:

$$\varepsilon^*(net_i) = \frac{\varepsilon(net_i) - \min(\varepsilon(net_i))}{\max(\varepsilon(net_i)) - \min(\varepsilon(net_i))}$$

--- (50)

$$\delta(net_i) = e^{-\psi\varepsilon^*(net_i)}$$

--- (51)

Here, Eq (50) is a MSE normalization operation applied on each MLP represented by a chromosome. Eq (51) is the actual fitness function. In Eq (51), ψ is a positive constant. A fine choice from our experience is to set ψ to 6.0.

3) Select parents for reproduction based on their fitness. A roulette wheel selection scheme is adopted as in [26, 27]. The population of current generation is mapped onto a roulette wheel, where each chromosome is represented by a space that proportionally corresponds to its fitness.

4) Apply search operators in conjunction with the crossover and/or mutation operators like arithmetical crossover operator [29] and a non-uniform mutation operator to parent chromosomes to generate offspring, which form the next generations.

Crossover operator: For the arithmetical crossover operator, it was assumed that $C_2 = (c_1^2, ..., c_i^2, ..., c_n^2)$ and $C_2 = (c_1^2, ..., c_i^2, ..., c_n^2)$ are two chromosomes that have been selected to apply the crossover operator, then two offspring, H_k, $k=1,2,$ are created according to the following equations:

$$H_k = (h_1^k, ..., h_2^k, ..., h_n^k) \quad k = 1, 2$$

$$h_i^1 = \lambda c_i^1 + (1-\lambda)c_i^2$$

$$h_i^2 = \lambda c_i^2 + (1-\lambda)c_i^1$$

--- (52)

Where λ is a user specified positive constant. In our experiments, λ is set to 0.28.

Mutation operator: Let us suppose $C = (c_1, ...c_2, ...c_n)$ is a parent chromosome, $C \in [a_i, b_i]$ is a gene to be mutated, and a_i and b_i are the lower and upper ranges for gene c_i. A new gene in the offspring chromosomes, c_i' may arise from the application of two different mutation operators respectively. The first mutation operator is single point random mutation, in which a single gene c_i' randomly chosen number from range $[a_i, b_i]$ to replace c_i and to form new chromosome C'.This mutation operator is sometimes called uniform mutation. Another mutation operator is the non-uniform mutation operator. Assuming this operator is applied in a generation t, and g_{max} the maximum numbers of generations are described as:

$$c_i' = \begin{cases} c_i + \Delta(t, b_i - c_i) & \text{if } \tau = 0 \\ c_i - \Delta(t, c_i - a_i) & \text{if } \tau = 1 \end{cases}$$

$$\Delta(t, y) = y\left(1 - r^{\left(1 - \frac{t}{g_{\max}}\right)^b}\right)$$

--- (53)

Where τ is a random binary number having value 0 or 1, and b is a parameter chosen by the user, which determines the degree of dependency on the number of iterations. This function gives a value in the range [0, y] such that the probability of returning a number close to zero increases as the algorithm advances. The size of the gene generation interval shall be lower with the passing of generations. In the algorithm, b is set to 0.5. The property of combining these crossover and mutation operators reduces the risk of premature convergence.

TABLE II
BENEFITS AND LIMITATIONS OF VARIOUS MACHINE LEARNING ALGORITHMS

Machine Learning Algorithm	Benefits	Assumptions and / or Limitations
Neural Network	•can be used for classification or regression •able to represent Boolean functions (AND, OR, NOT) •tolerant of noisy inputs •instances can be classified by more than one output	•difficult to understand structure of algorithm •too many attributes can result in over fitting •optimal network structure can only be determined by experimentation
Support Vector Machine	•models nonlinear class boundaries •over fitting is unlikely to occur •computational complexity reduced to quadratic optimization problem •easy to control complexity of decision rule and frequency of error	•training is slow compared to Bayes and Decision trees •difficult to determine optimal parameters when training data is not linearly separable •difficult to understand structure of algorithm
Fuzzy logic	• different stochastic relationships can be identified to describe properties	•Priori knowledge is very important to get good results. •precise solutions are not obtained if the direction of decision is not clear.
Genetic Algorithm	•can be used in feature classification and feature selection •primarily used in optimization always finds a "good" solution (not always the best solution) • can handle large, complex, non differentiable and multimodal spaces. • Efficient search method for a complex problem space. • good at refining irrelevant and noisy features selected for classification.	•computation or development of scoring function is nontrivial •not the most efficient method to find some optima, rather than global •complications involved in the representation of training/output data

No vertical lines in table. Statements that serve as captions for the entire table do not need footnote letters.
[a]Gaussian units are the same as cgs emu for magnetostatics; Mx = maxwell, G = gauss, Oe = oersted; Wb = weber, V = volt, s = second, T = tesla, m = meter, A = ampere, J = joule, kg = kilogram, H = henry.

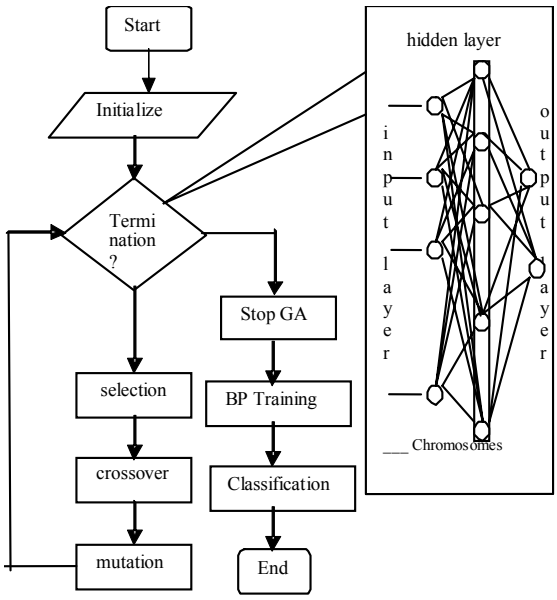


Figure 8:Classification using combined Neural Network and GA

2) Weight connections optimization using conjugate gradient descent algorithm

Genetic algorithm is used effectively in the evolution to find a near-optimal set of connection weights globally without computing gradient information and without weight connections initialization. Hence a local search procedure into the evolution using the conjugate gradient descent algorithm to find the best connection weights at the local error surface is incorporated. This procedure is completed by applying a BP algorithm on the GA established initial connection weights. The overall framework of the proposed method could be summarized as Figure 8. First, at the initialization stage, the neural network structure, including number of input nodes, hidden nodes, and output nodes are specified according to the specific classification application. Connection weights corresponding to this MLP structure are encoded in GA's chromosomes; each chromosome represents one MLP structure with given connection weights contained in its genes. Second, at the GA-based weight connection training stage, these initialized chromosomes which may belong to different populations are evolved generation to generation by using GA according to the fitness and MSE performance of the correspondent MLP. Finally, at the stage of local optimization of the error surface with BP, best connection weights matrix contained in the chromosome with best fitness is chosen.

Therefore, genetic Algorithms handle large, complex, non differentiable and multi model spaces for image classification and many other real world applications.

TABLE III
COMPARATIVE ANALYSIS OF DIFFERENT IMAGE CLASSIFICATION TECHNIQUES WITH RESPECT TO VARIOUS PARAMETERS

Parameter	Artificial Neural Networks	Support Vector Machines	Fuzzy logic	Genetic Algorithms
Type of approach	Non-parametric	Non-parametric with binary classifier	Stochastic	Large time series data
Non-linear decision boundaries	Efficient when the data have only few input variables.	Efficient when the data have more input variables.	Depends on priori knowledge for decision boundaries.	Depends on the direction of decision
Training speed	Network structure, momentum rate, learning rate, converging criteria.	Training data size, kernel parameter, class separability	Iterative application of the fuzzy integral	Refining irrelevant and noise genes
Accuracy	Depends on number of input classes.	Depends on selection of optimal hyper plane.	Selection of cutting threshold	Selection of genes
General performance	Network structure.	Kernel parameter	Fused fuzzy integral	Feature selection

3. COMPARATIVE ANALYSIS OF VARIOUS MACHINE LEARNING ALGORITHMS

The advanced image classification techniques like artificial neural networks, support vector machines, fuzzy logic, genetic algorithms and their combination are analyzed and compared with respect to several parameters. The benefits and limitations of these classification techniques are as given in Table II. Artificial neural networks have the advantages mainly of more tolerance to noise inputs and representation of boolean function apart from others. But too many attributes may result in over fitting. In support vector machines over fitting is unlikely to occur. The computational complexity and complexity of decision rule are reduced in SVM. Fuzzy measures have the benefit of identification of various stochastic relationships to describe the properties of the image. But priori knowledge is very important to get good results. Genetic algorithms are primarily used in optimization and always have a good solution. But the computation of scoring function is non trivial.

The comparative analysis of different image classification algorithms with respect to several parameters is as given in Table III. The artificial neural networks and support vector machines follows non-parametric approach whereas fuzzy measures use stochastic properties for image classification. The selection of non-linear boundary is efficient when the data have only few input variables in ANN and vice versa in SVM. In fuzzy logic it depends on priori knowledge whereas in genetic algorithms it depends on the direction of decision. The training speed in the neural networks depends on network structure, momentum rate, learning rate and converging criteria. In SVM it depends on training data size and class separability. Fuzzy logic incorporates the training speed depending on the isolation of the relevant information by iterative application of the fuzzy integral. The training speed could be improved by refining irrelevant and noisy genes in genetic algorithms. Along with these the parameters accuracy and general performance are tabulated in Table III.

4. CONCLUSIONS

This paper attempts to study and compare artificial neural networks and other methods of machine learning algorithms for image classification. The study concluded that the neural network approach of classification improves the accuracy and the finer information from the individual class is obtained by using textures. This study emphasizes that that kernel type and kernel parameter affect the shape of the decision boundaries as located by the SVM and thus influence the performance of the SVM. It is found that the optimum results are obtained if the stochastic information is fused by the fuzzy integral. Fuzzy support vector machines resolve unclassifiable regions caused by conventional support vector machines and its generalization ability is superior. The combined genetic algorithm plus conventional classifier system achieves higher performance than either the conventional classifier or the GA alone. This paper has showed that carefully designed genetic algorithm-based neural network outperforms gradient descent-based neural network. This has been supported by the analysis of the changes of connection weights and biases of the neural network. The neural network topology

described in this study is determined manually. A substitute method is to apply the genetic algorithm for neural network structure optimization, which will be a part of the future work.

REFERENCES

[1] S Haykin, *Neural Network - a Comprehensive Foundation; a Computational Approach to Learning and Machine Intelligence*, Macmillan, NY, 1994.

[2] J. M. Zurada, *Introduction to Artificial Neural Networks System*. Jaico Publishing House.

[3] Freeman, "Artificial Neural Network Algorithm" *Applications and Programming*, Comp and Neural Systems Series, 1990.

[4] A. Kulkarni, *.Artificial Neural Network for Image Understanding, Van* Nostrand Reinhold, New York, 1994.

[5] J. Anderson, *An Introduction to Neural Network*.

[6] T Jackson*, Neural Computing,*. An Introduction.

[7] Haralick and Shanmugan, "Textural Features for Image Classification" .IEEE Transactions on Systems, Man and Cybernatics, vol . SMC 3, no 6, pp. 610, November 1973.

[8] V.N.Vapnik, *T he Nature of Statistical L earning T heory,* (New York: Springer-Verlag), 1995.

[9] J V. N Vapnik,, *Statistical L earning T heory* (New York: Wiley), 1998.

[10] C. J. C. Burges, "A tutorial on support vector machines for pattern recognition", *Data Mining and Knowledge Discovery*, **2**, 1998, pp. 121–167.

[11] R. K. Sundaram, *A First Course in Optimization Ttheory,* (New York: Cambridge University Press), 1996.

[12] B. Campbell, "Spatial correlation effects upon accuracy of supervised classification of land cover", *Photogrammetric Engineering and Remote Sensing*, 47, pp. 355–363, 1981.

[13] P. J .Bickel, C. A. J. Klaassen, Y. Ritov and J. A. Wellner," Efficient and Adaptive Estimation for Semiparametric Models", The Hopkins Univ. Press, Baltimore and London. 1993,

[14] S. P Banks, 1990**,** "Signal Processing, Image Processing and Pattern Recognition", Prentice Hall, New York.

[15] J. Teubner, *Digital Image Processing*, Prentice Hall, New York, 1993.

[16] H. Hetzheim, "Using Martingale Representation to Adapt Models for Non-Linear Filtering", Proceedings of ICSP 93, International Academic Publ., Beijing , pp 32-35, 1993.

[17] H. Hetzheim, "Anwendung der Fuzzy-Logic in der Bildverarbeitung," Vision Jahrbuch' 93, Sprechsaal Pub., Coburg, pp. 47-53, 1993.

[18] M. H. Hetzheim, "Detection of stochastic structures in Images by Fuzzy and Choquet Integrals", ACCV'95, Second Asian Conference on Computer Vision ,III , pp 116-120,1993.

[19] Sugeno, "Theory of fuzzy integrals and its application", Doctoral Thesis, Tokyo Institute of Technology, (1974).

[20] U. H.G. KreBel, "Pairwise Classification and Support Vector Machines," in B. Scholkopf, C. J. C. Burges, and A. J. Smola, Editors, "*Advances in Kernel Methods: Support Vector Learning",* pp 255-268, The MIT Press, Cambridge, MA, 1999.

[21] J. Richards and X. Jia, Remote Sensing Digital Image Analysis, Springer, New York, 3rd ed., 1999.

[22] W. Banzhaf, P. Nordin, R. E. Keller, and F. D. Francone, Genetic Programming: An Introduction, Morgan Kaufmann, San Francisco, CA, 1998.

[23] F. Francone, M. Conrads, W. Banzhaf, and P. Nordin, "Homologous crossover in genetic programming," in Proc. of Genetic and Evolutionary Computation Conference (GECCO'99), B. et al., ed., vol. 2, pp. 1021–1026, Morgan Kaufmann, San Francisco, CA, 1999.

[24] N. Harvey, S. Brumby, S. Perkins, R. Porter, J. Theiler, A. Young, J. Szymanski, and J. Bloch, "Parallel evolution of image processing tools for multispectral imagery," in Proc. SPIE 4132, (San Diego, CA), July 2000.

[25] D. E. Rumelhart, G. E .Hinton, and R.J. Williams, Learning representations by back-propagating errors. Nature, 323: 533-536, 1986.

[26] D.E Goldberg, *Genetic Algorithms in Search, Optimization*, and Machine Learning. Addison-Wesley, New York, 1989.

[27] J.H Holland, *Adaptation in Natural and Artificial Systems*, The University of Michigan Press, 1975.

[28] Herrera F., Lozano M., Verdegay J. L., Tackling Real-Coded Genetic Algorithms:

Operators and Tools for Behavioral Analysis. NEC Research Index. http://citeseer.nj.nec.com/

[29] Michalewicz Z., 1992. Genetic Algorithms + Data Structures = Evolution Programs. Springer-Verlag, New York.