NASA-CR-194291

Research Institute for Advanced Computer Science
NASA Ames Research Center

IN-63-CR
185435
49 P

A Review and Analysis of Neural Networks for Classification of Remotely Sensed Multispectral Imagery

Justin D. Paola
Robert A. Schowengerdt

RIACS Technical Report 93.05

June 1993

Submitted to Remote Sensing of Environment

A Review and Analysis of Neural Networks for Classification of Remotely Sensed Multispectral Imagery

Justin D. Paola
Robert A. Schowengerdt

The Research Institute for Advanced Computer Science
is operated by Universities Space Research Association,
The American City Building, Suite 212, Columbia, MD 21044, (410) 730-2656

Work reported herein was partially supported by the National Aeronautics and Space Administration under Contract NAS 2-13721 to the Universities Space Research Association (USRA) and under Grant NAG 5-2198 to the University of Arizona Department of Electrical and Computer Engineering.

ABSTRACT

A literature survey and analysis of the use of neural networks for the classification of remotely sensed multispectral imagery is presented. As part of a brief mathematical review, the backpropagation algorithm, which is the most common method of training multi-layer networks, is discussed with an emphasis on its application to pattern recognition. The analysis is divided into five aspects of neural network classification: 1) input data preprocessing, structure, and encoding, 2) output encoding and extraction of classes, 3) network architecture, 4) training algorithms, and 5) comparisons to conventional classifiers. The advantages of the neural network method over traditional classifiers are its non-parametric nature, arbitrary decision boundary capabilities, easy adaptation to different types of data and input structures, fuzzy output values that can enhance classification, and good generalization for use with multiple images. The disadvantages of the method are slow training time, inconsistent results due to random initial weights, and the requirement of obscure initialization values (e.g., learning rate and hidden layer size). Possible techniques for ameliorating these problems are discussed. It is concluded that, although the neural network method has several unique capabilities, it will become a useful tool in remote sensing only if it is made faster, more predictable, and easier to use.

INTRODUCTION

Researchers from many disciplines are interested in the use of remotely sensed data to enhance their research. The focus of this paper is the use of this data to produce ground cover classifications. Standard classification methods usually require assumptions about the underlying statistics of the data, the most common being that the data for each ground cover class is Gaussian distributed (e.g., Richards, 1986; Schowengerdt, 1983; Swain, 1978). If these assumptions turn out to be correct then the statistical classifier is the optimal choice for the problem. In recent years the artificial neural network, or multi-layer perceptron, has been developed and applied to general pattern recognition problems. Neural network classifiers are non-parametric and may be more robust when distributions are strongly non-

Gaussian (Lippmann, 1987). Because of this capability, researchers have begun investigating the use of this method for classifying remotely sensed multispectral imagery.

There are two stages involved in using neural networks for multispectral image classification: the training stage and the classification stage. Training data is similar to that used in a statistical classifier. The network is trained, typically by the backpropagation algorithm, until some targeted minimal error is achieved between the desired and actual output values of the network. Once training is complete, the network is used as a feed-forward structure to produce a ground cover classification for the entire image.

Since research on the use of neural networks for the classification of multispectral imagery has just begun within the last four years, no comprehensive study of this process has been presented. The goal of this paper is to present a thorough analysis of the technique, discussing all aspects of the process from selection of input data to final assignment of classes, and to review work currently appearing in the literature. Particular emphasis is given to both the unique problems and capabilities of the neural network method as compared to standard classifiers. A general review of neural networks is presented first. Following this is the analysis and literature survey, organized according to topic. At the end, conclusions are drawn about the capabilities, problems, and potential future use of neural networks as a tool of remote sensing.

1. A REVIEW OF NEURAL NETWORKS

Network structure

The basic element of a neural network is the processing node (Figure 1). Each processing node mimics the biological neuron and performs two functions. First it sums the values of its inputs. This sum is then passed through an arbitrary "activation" function to produce the node's output value. The processing nodes are organized into layers, each generally fully interconnected to the following layer. There are no interconnections within a layer, however. In addition, there is an input layer that serves as a distribution structure for the data being presented to the network. No

The Processing Node

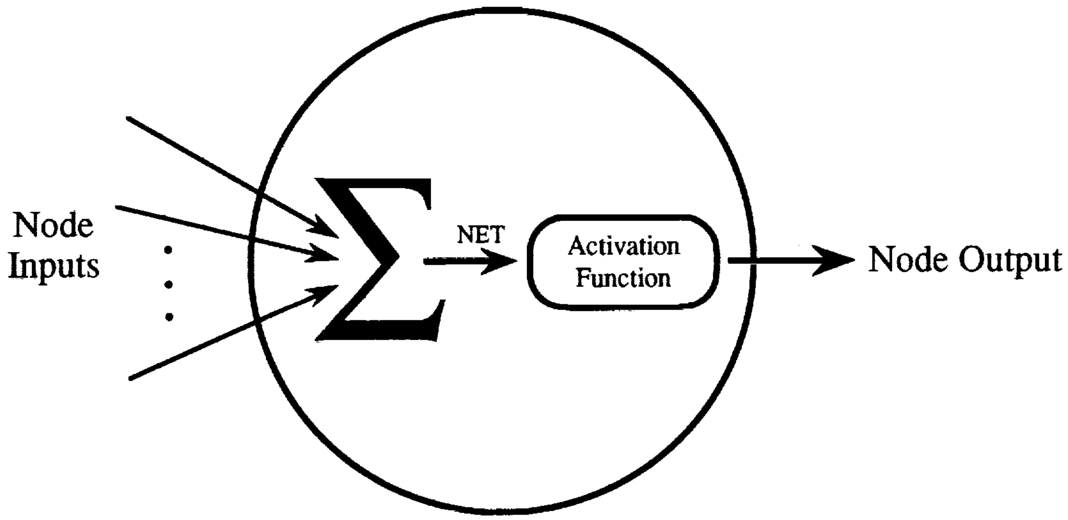


Figure 1: Internal structure of a neural network processing node. Node inputs are summed and passed through an activation function which is usually a sigmoid. The value of the sum, NET, is referred to in the discussion of backpropagation.

processing is done at this layer. One or more actual processing layers follow the input layer. The final processing layer is called the output layer. Any layers in between the input and output layers are termed "hidden" layers. Figure 2 shows the generic structure for a commonly used configuration, the three layer neural network. The interconnections between each node have an associated weight. When a value is passed down that interconnection, it is multiplied by the weight. These weight values contain the distributed knowledge of the network.

Backpropagation training algorithm

Until recently, there were no effective algorithms for adjusting the interconnecting weight values for minimal overall training error in multi-layer networks (Lippmann, 1987). The generalized delta rule, or backpropagation, presented in 1986 by Rumelhart *et al.*, is now one of the most commonly used methods. This is an iterative, gradient descent training procedure. The training data consists of a pair of data vectors. The input data vector is the pattern to be learned and the desired output vector is the set of output values that should be produced by the network upon recall of the input training pattern. The goal of the training is to minimize the overall error between the desired and actual outputs of the network.

Because of its widespread use we will provide a brief description of the backpropagation algorithm. The following equations were adapted from Pao (1989) and Rumelhart *et al.* (1986), and a full derivation can be found in these sources. Refer to Figure 2 for notation. The error for one input training pattern, t, is a function of the desired output vector, d, and the actual output vector, o, given by

$$E = \frac{1}{2} \cdot \sum_k (d_k - o_k)^2. \tag{1}$$

The value produced by output node k, o_k, is the activation function, f, evaluated at the sum produced within node k, NET_k (Figure 1). NET_k, in turn, is a function of the weights between the hidden and output layer, w_{kj}, and the outputs of the hidden layer nodes, o_j:

$$o_k = f(NET_k) = f\left(\sum_j (w_{kj} \cdot o_j)\right). \tag{2}$$

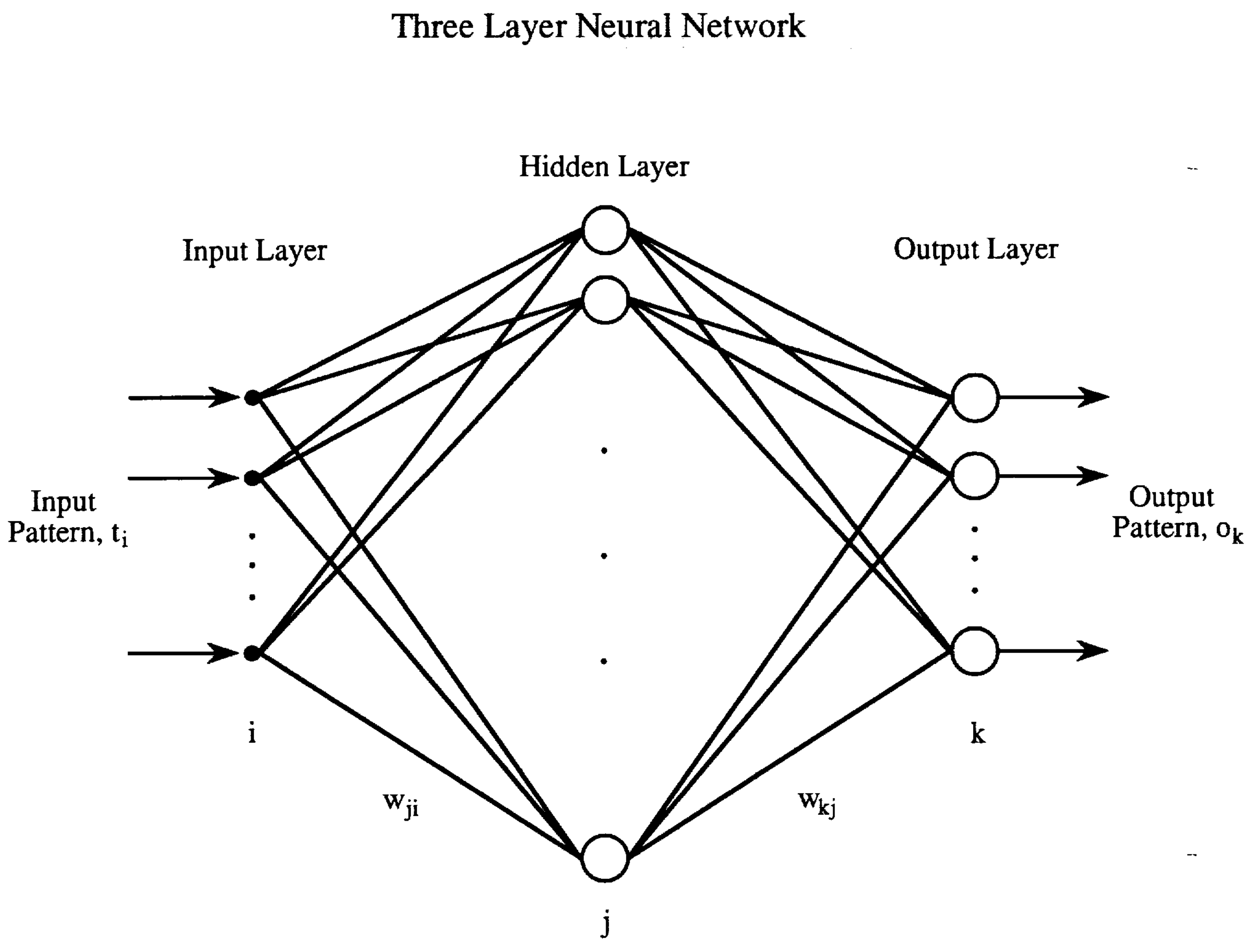


Figure 2: A generic three layer neural network structure showing weight values and layer labels referred to in the discussion of the backpropagation algorithm.

To minimize the error, the change from one iteration to the next in w_{kj} must be proportional to the derivative of the error with respect to the weights summed over all training patterns. The new weights to be used in iteration (m + 1) can be expressed as a function of quantities from the current iteration (m) as

$$w_{kj}(m+1) = w_{kj}(m) - LR \cdot \sum^{P} \frac{\partial E(m)}{\partial w_{kj}(m)}, \tag{3}$$

where LR, the learning rate, is the percentage of the step taken towards the minimum error in each iteration. Substituting for $\frac{\partial E}{\partial w_{kj}}$ (see Pao, 1989; Rumelhart *et al.*, 1986), the weight change formula becomes

$$w_{kj}(m+1) = w_{kj}(m) + LR \cdot \sum^{P} \left((d_k - o_k) \cdot f'(NET_k) \cdot o_j \right). \tag{4}$$

To determine the weight change formula for any previous sets of weights, such as w_{ji} in Figure 2, the error must be propagated back from the output layer. The error term at a hidden layer node is computed from the sum over all variations in the output layer. This results in a more complex weight change formula for weight w_{ji},

$$w_{ji}(m+1) = w_{ji}(m) + LR \cdot \sum^{P} \left(f'(NET_j) \cdot \sum_{k} \left((d_k - o_k) \cdot f'(NET_k) \cdot w_{kj} \right) \cdot t_i \right), \tag{5}$$

where NET_j is the internal sum of hidden layer node j. It can be seen that part of this equation is contained in the previous weight change formula (eq. 4). This carry over of terms from one weight change equation to the next represents the backpropagation of the error.

The equations are easily extended to the case of more than one hidden layer. The update equation for each set of weights before w_{kj} is a function of quantities calculated for the previous set of weights. As each training pattern is presented, the relevant quantities are summed at each processing node. The total error between desired and actual outputs is also summed. If this error is still above some predetermined threshold when the training cycle is completed, the weights are adjusted and training continues.

Sigmoid activation function

Any differentiable function may be used for the activation function, f. The most common is the sigmoid function (Figure 3), defined as

$$f(\mathrm{NET}) = \frac{1}{1+e^{-\mathrm{NET}}}, \tag{6}$$

where NET is the sum of weighted inputs to the processing node (Figure 1). The shape of the sigmoid can be modified by multiplying NET by a constant (Pao, 1989). However, the exact features of the curve are less important than the general S-shape, and the overall network will function just as well with or without this change (Caudill, 1988). One key advantage of the sigmoid function is that its derivative can be expressed in terms of the function itself, i.e.,

$$f'(\mathrm{NET}) = f(\mathrm{NET})\cdot(1-f(\mathrm{NET})). \tag{7}$$

Since the function has already been computed during forward propagation of the training data, the computation time of backpropagation is reduced over the general case.

Some features of the sigmoid activation function are important to network performance. Zero and one values are possible only with inputs of $\pm\infty$. To account for this, values of 0.1 and 0.9 are generally used to represent the low and high values of binary numbers (Pao, 1989; Rumelhart *et al.*, 1986). The activation function has a nearly linear input/output relationship in between these two extreme values. But as the outputs of a node approach these values, the derivative of the activation function decreases to zero, thus ensuring that only very small changes will occur in the weights (Caudill, 1988). The derivative has a maximum value when the output is 0.5. Since the change in weights is proportional to the derivative value, the weights will change rapidly in this case and help influence the node to commit to a high or low value. This feature probably contributes to the stability of the learning stage (Rumelhart *et al.*, 1986).

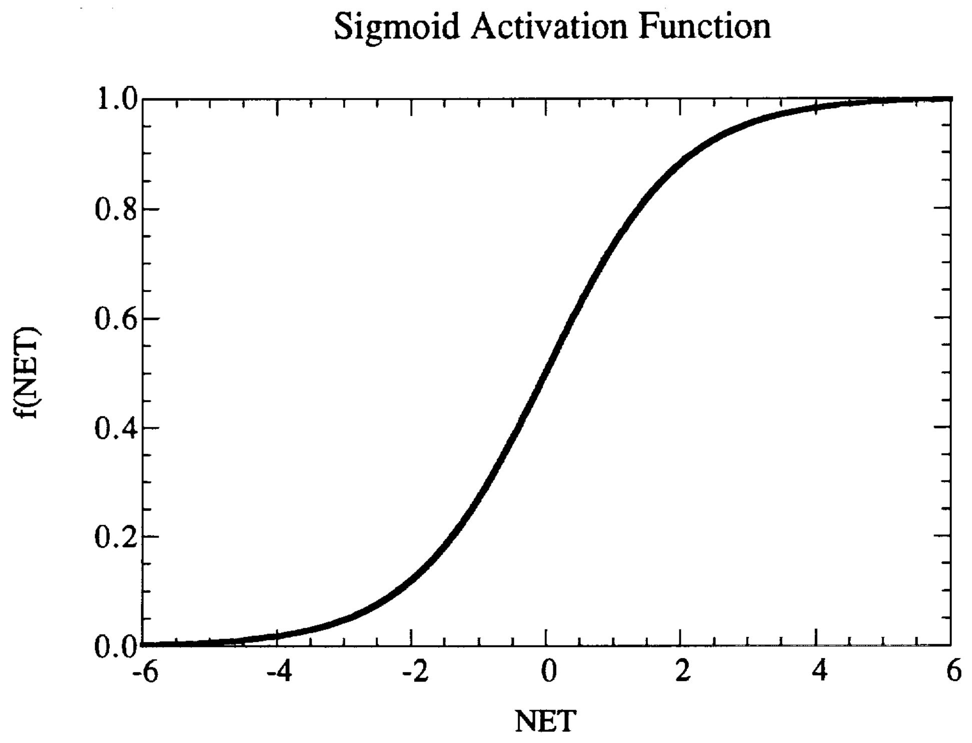


Figure 3: The sigmoid activation function.
NET is the weighted sum of the inputs to the processing node.

Other neural network training considerations

Once the training algorithm and activation function have been determined, the next issue is when to update the weights. The backpropagation algorithm provides weight change terms that should be summed for all training patterns to obtain a true gradient descent of the overall training error. If weight corrections are carried out after each training pattern presentation, the method is not a true gradient searching procedure (Pao, 1989). Many neural network implementations, however, employ this technique because the code for this method, sometimes called "sequential" training, is simpler. It also allows for more flexibility with the training data since all patterns do not have to be presented before weight changes occur. However, besides the disadvantage of not obtaining a true gradient descent, this method is hindered by additional processing time per training cycle (which constitutes one presentation of each pattern) because the weights are adjusted with every training pattern instead of only at the end of the cycle. It is also less amenable to a parallel implementation. The method of summing the changes for all training patterns and updating the weight values at the end is referred to as "batch" or "epoch" training.

Training a neural network involves setting several initial parameters. The first step is to determine the training data and corresponding desired outputs for that training data. Then overall network structure must be defined. The weight updating method (sequential or batch) must be chosen for the training algorithm. When the training process begins, all of the weights of the network must be set to random values, because it is not possible to obtain a set of unequal weights containing the distributed knowledge of the network if a set of equal weights is used for the initial configuration (Rumelhart *et al.*, 1986). Then the learning rate parameter must be set, generally by trial and error. Adaptive learning rates are one way to avoid this trial and error process. One adaptive strategy is to adjust the learning rate downward after some training interval if the overall training error has increased and upward if the overall error has decreased. A similar method has been employed by Heermann and Khazenie (1992). With this method the initial value of the learning rate is not crucial to the success of the training and training speed is increased since the learning rate is adjusted to the highest value that does not cause instability.

The final parameter is the training convergence criterion. Only in simple cases is it possible to train the network to zero training error. Thus, some criterion for terminating the training process must be established, such as the mean square error falling below a specified threshold. When this criterion is met the network training is complete and the network may be used as a feed-forward classifier. The exact value of this threshold is another parameter that must be determined experimentally. This is a problem of generalization versus specialization: if the network is trained too well on the training data it might not function accurately on the rest of the image; on the other hand, if it is not trained well enough it will not be able to separate the classes, even in the training data, to an acceptable extent. The choice of a threshold also might be influenced by training time. A disadvantage of the neural network method is the enormous length of time often required to obtain the minimum training error. A higher threshold value can be used to keep the network training time reasonable.

In addition to long training time, another drawback of neural networks is that backpropagation, like all gradient descent algorithms, is not guaranteed to find the global minimum error. During the training phase, the network takes the steepest descent from the current position to one of lower error (Caudill, 1988). If the network encounters a valley, or local minimum, in the error space, it can become stuck and the error will not decrease to the global minimum value. It is also possible for the system to oscillate between two points. One way to alleviate these problems is to add some fraction of the weight change calculated in the previous iteration, $\alpha \cdot \Delta w_{kj}(m)$, to the weight update formula $w_{kj}(m+1)$ (eq. 4) (Pao, 1989). The added push from this previous weight change term hopefully is enough to keep the network from becoming stuck in local minima during training. The momentum parameter, α, like the learning rate, is set at the beginning of the training and must be determined experimentally. It is possible to implement an adaptive momentum term as well (Heermann and Khazenie, 1992).

Neural networks for pattern recognition

Neural networks have been applied to a wide range of problems, including pattern recognition. Before discussing the particular case of classification of multispectral imagery, it is helpful to examine the general pattern recognition capabilities of neural networks. During training, the network weights are adjusted in an attempt to minimize the error between desired and actual outputs for the training patterns[1]. This results in the formation of optimal decision boundaries in the feature space that determine class membership. The advantage of the neural network method is that this process is done with no assumptions about class distribution, hopefully leading to a more general technique.

One way to view the operation of a neural network is to consider how the feature space is partitioned for each output. Lippmann (1987) provides an excellent discussion of the decision region capabilities of networks as a function of the number of layers. His discussion focuses on a two class case using multi-layer perceptrons with one output and discontinuous, hard limiting (+1 for positive inputs and -1 for negative inputs) activation functions. A two layer network (no hidden layer) can form hyperplane decision regions in the feature space (Figure 4). A three layer network can produce any, possibly unbounded, convex region[2] in the input data space. The nodes in the hidden layer produce a series of hyperplane decision regions as in the output layer of the two layer network. Each node of the output layer then performs a logical AND operation on these hyperplanes to produce the final convex decision regions for that output. With the addition of a second hidden layer, any form of decision region can be created. The output layer in this case takes the set of convex decision regions produced by the previous layer and performs a logical OR operation, thus creating arbitrarily shaped decision regions. The advantage of the four layer network is that it can form regions that are disconnected. Since the four layer net can produce arbitrarily complex decision regions there is no need for the use of any higher order nets.

In order to be used for the classification of multispectral imagery, the network must be able to discern multiple classes, not just separate two classes as shown in Figure 4. Lippmann (1987) states that the behavior of the required type of net (i.e., one with multiple outputs and the sigmoid activation function) is similar to the single output, hard limiting activation function case of Figure 4. The analysis of this net is more difficult, however,

[1] Note that, as with the maximum-likelihood classifier, the goal is to minimize the classification error of the training data.

[2] A convex region is one in which a line between any two points of the region will be contained entirely within that region.

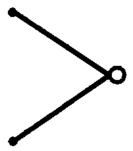
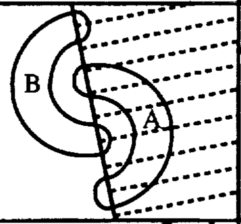
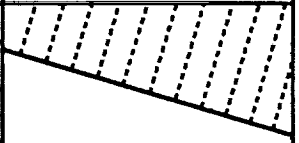
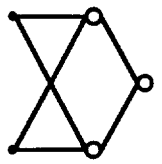
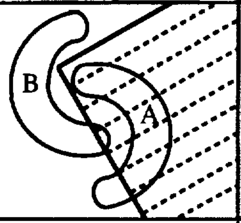
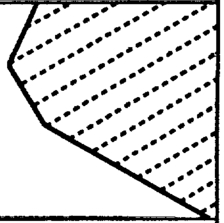
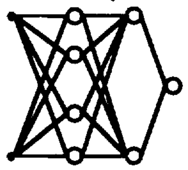
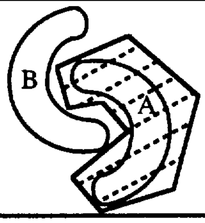
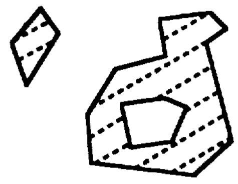
Network Structure	Type of Decision Regions	Classes with Meshed Regions	Most General Region Shapes
Two layer 	Half plane bounded by hyperplane		
Three layer 	Convex open or closed regions		
Four layer 	Arbitrary (Complexity limited by number of nodes)		

Figure 4: Types of decision regions that can be formed in the input data space by two, three, and four layer neural networks with hard limiting activation functions and one output node. Regions for networks with sigmoid activation functions and multiple outputs will be more smoothed but have similar properties. (After Lippmann, 1987)

because the continuous, non-linear activation function results in decision regions that are usually bounded by smooth curves.

Another way to view the operation of the neural network is as a non-linear transformation of the feature space into a new space in which the data is linearly separable (Pao, 1989). The nodes of the final layer, in this case, perform a simple hyperplane resolution on the output space of the previous layer. The complexity of the separation determines the number of layers needed.

2. APPLICATIONS OF NEURAL NETWORKS TO MULTISPECTRAL IMAGES

The ability of the neural network to produce a classification with arbitrarily shaped decision regions without any prior knowledge of the statistical distribution of that data makes it a promising candidate for the classification of remotely sensed multispectral imagery. Many researchers have addressed various aspects of this technique. This comprehensive analysis and literature survey is organized into several topical sections:

- Input data preprocessing, structure and encoding
- Output encoding and extraction of classes
- Network architecture
- Training algorithms
- Comparisons to conventional classifiers

2.1 Network input

Preprocessing

The first step in using neural networks for the classification of multispectral imagery is to determine the type and form of input data to be fed into the network. The image data that has been most commonly used is Landsat Thematic Mapper (Bischof *et al.*, 1992; Heermann and Khazenie, 1992; Kiang, 1992; Civco, 1991; Cromp, 1991; Liu and Xiao, 1991; Mulder and Spreeuwers, 1991; Hepner *et al.*, 1990; Ritter and Hepner, 1990; McClellan *et al.*, 1989), but some studies have involved SPOT (Dreyer, 1993; Wilkinson *et* *al.*, 1992; Kanellopoulos *et al.*, 1991), Landsat MSS (Cromp, 1991; Benediktsson *et al.*, 1990a), simulated HIRIS (Benediktsson *et al.*, 1990b), merged AVHRR and SMMR (Key *et al.*, 1990; Key *et al.*, 1989), and digitized aerial photography (Medina and Vásquez, 1991).

As with any classification routine, preprocessing, or feature extraction, can be carried out on the image data to condense the data and help the neural network differentiate the classes. Principal components analysis is often used to reduce dimensionality in classification problems (e.g., Schowengerdt, 1983). Beyond the initial phase of data reduction, however, additional non-linear processing of the data before presenting it to the network can result in linearly separable classes that dramatically decrease training time over the non-linearly separable case (Pao, 1989). Three of the papers in this survey included discussion of preprocessing. Dreyer (1993) calculated a number of textural features based on gray-level statistics for image segments used in training. He found the use of these features increased the accuracy of a "field" class, had no effect for "urban" and "water" classes and actually decreased the accuracy of a "forest" class. Key *et al.* (1990) also used texture calculations such as second moment and entropy to produce a single texture measure for each pixel in the classification of land cover and cloud types in the Arctic. Classification results in this case were superior to those that used spectral pixel values only. Civco (1991), in a standard land cover classification, presented the network with a single mean vector for each class of the training data. This resulted in a drastic reduction of the training set size but forced a statistical measure into the training process. He points out that this is not the optimal way to implement a neural network classifier, as a measure of randomness needs to be presented to the network during training to account for the variability of the data during classification.

Another potential use of preprocessing routines might be for the smoothing of noise in the image data. The use of a 3x3 spatial mean and root mean square difference relative to the center pixel as input, for example, both introduces texture and masks dropped or bad pixels in the training data that, since they are not representative of the desired class, might cause training time to increase. In general, though, given the original image data and enough training time, the neural network can perform any necessary feature extraction transparently through the adjustment of its weights during backpropagation. Although a speed advantage can be gained from external preprocessing of the data, the network will be more generally applicable if it is presented with just the raw data. For instance, presenting the network with the entire window of pixels used to calculate texture instead of the single preprocessed texture value will give the network more flexibility. The network can use the additional pixel values as necessary to help separate the classes.

Structure

The simplest structure for data input (also used in most statistical classifiers) is that for reading one multispectral pixel into the network. One input node or a set of input nodes is used to represent the data for each spectral band (Figure 5). This is the favored input technique (e.g., Dreyer, 1993; Bischof *et al.*, 1992; Kiang, 1992; Li and Si, 1992; Wilkinson *et al.*, 1992; Cromp, 1991; Kanellopoulos *et al.*, 1991; Liu and Xiao, 1991; Medina and Vásquez, 1991; Short, 1991; Benediktsson *et al.*, 1990a; Benediktsson *et al.*, 1990b; Key *et al.*, 1990; Key *et al.*, 1989; McClellan *et al.*, 1989). A natural extension is to use a 3x3 window of pixel data from each band of the image as the input (Figure 6). This has the advantage of introducing texture information into the training procedure (Hepner *et al.*, 1990; Ritter and Hepner, 1990), but results in greatly increased training time per cycle because of the nine-fold increase in input pixels. A compromise is to use a 3x3 window in one band and only the center pixel of the window in the rest of the bands (Bischof *et al.*, 1992).
Another feature of neural networks is that additional sources of data can be added to the classification procedure by simply adding input nodes. This can be done with statistical classifiers by adding another "band" to the image data. The advantage of the neural network method is that one does not need to consider the distribution of the ancillary data, which will often be different from that of the multispectral data. The addition of a preprocessed version of the multispectral data such as the texture calculation used by Dreyer (1993) and Key *et al.* (1990) is one example of adding inputs to the network beyond those for the multispectral image itself. But other, unrelated, data sources can be used as well. Benediktsson *et al.* (1990a) added inputs for elevation, slope, and aspect data. The temptation to add input nodes blindly into the

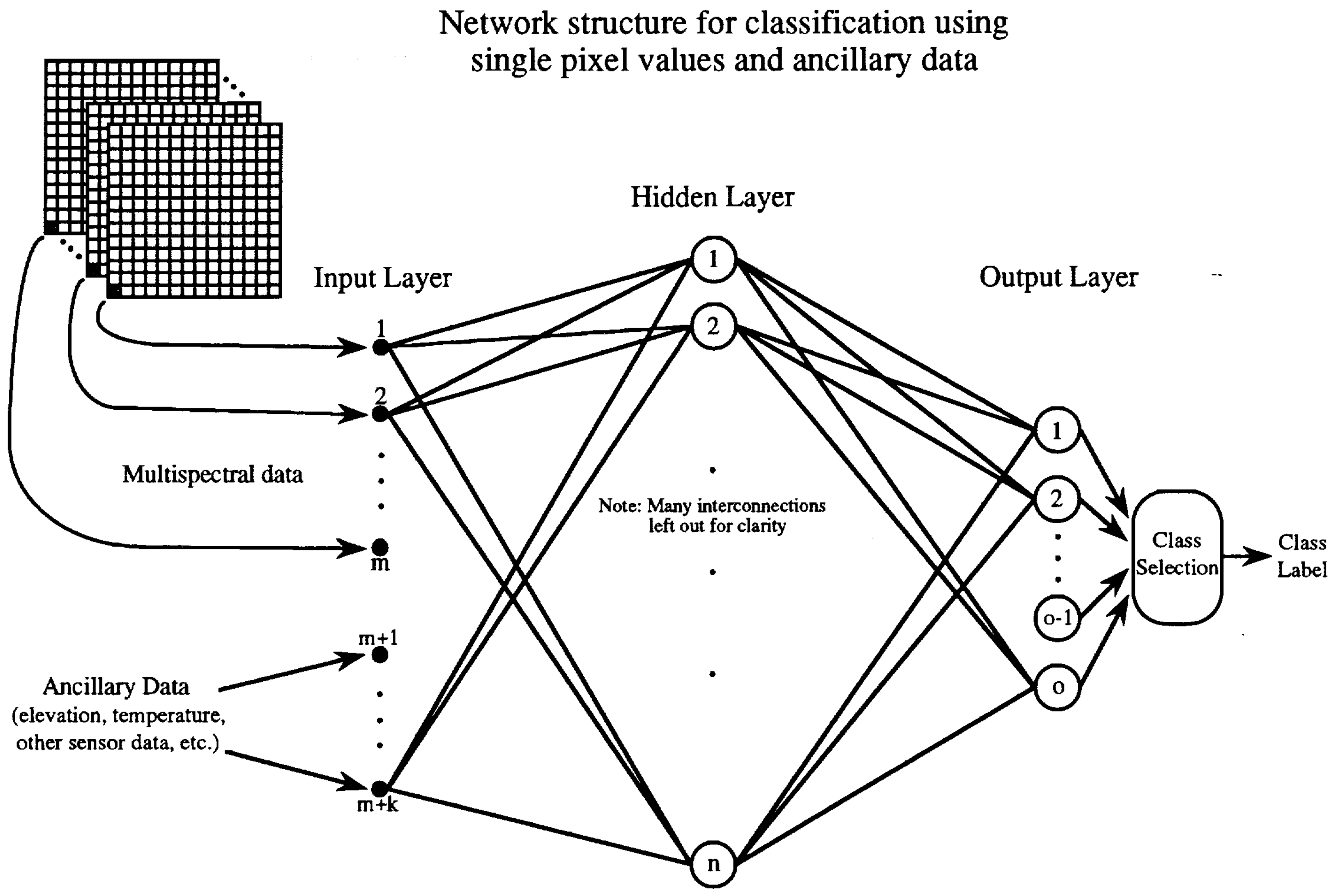

Network structure for classification using single pixel values and ancillary data
Hidden Layer
Input Layer
Output Layer
1
2
m
m+1
m+k
Multispectral data
Ancillary Data (elevation, temperature, other sensor data, etc.)
Note: Many interconnections left out for clarity
n
o-1
o
Class Selection
Class Label

Figure 5: This simple three layer backpropagation neural network structure is one of the most commonly seen in the literature. The value of a pixel in each of *m* bands of the multispectral image is presented to the input layer along with data for that pixel from *k* other sources (if desired). These inputs are fanned out to the first processing layer, the hidden layer. The number of nodes in the hidden layer is arbitrary and is usually chosen by experimentation. The outputs of the hidden layer are in turn fanned out to the final processing layer, the output layer. In this case each output is used to represent one of *o* possible ground cover classes. When the network is used in feed forward mode for classification, the output values are usually continuous and class membership can be determined by a threshold or by choosing the highest output value. The output values can also be viewed as fuzzy membership values for each class. A measure of class mixing and the uncertainty of classification can be extracted from this set of values.

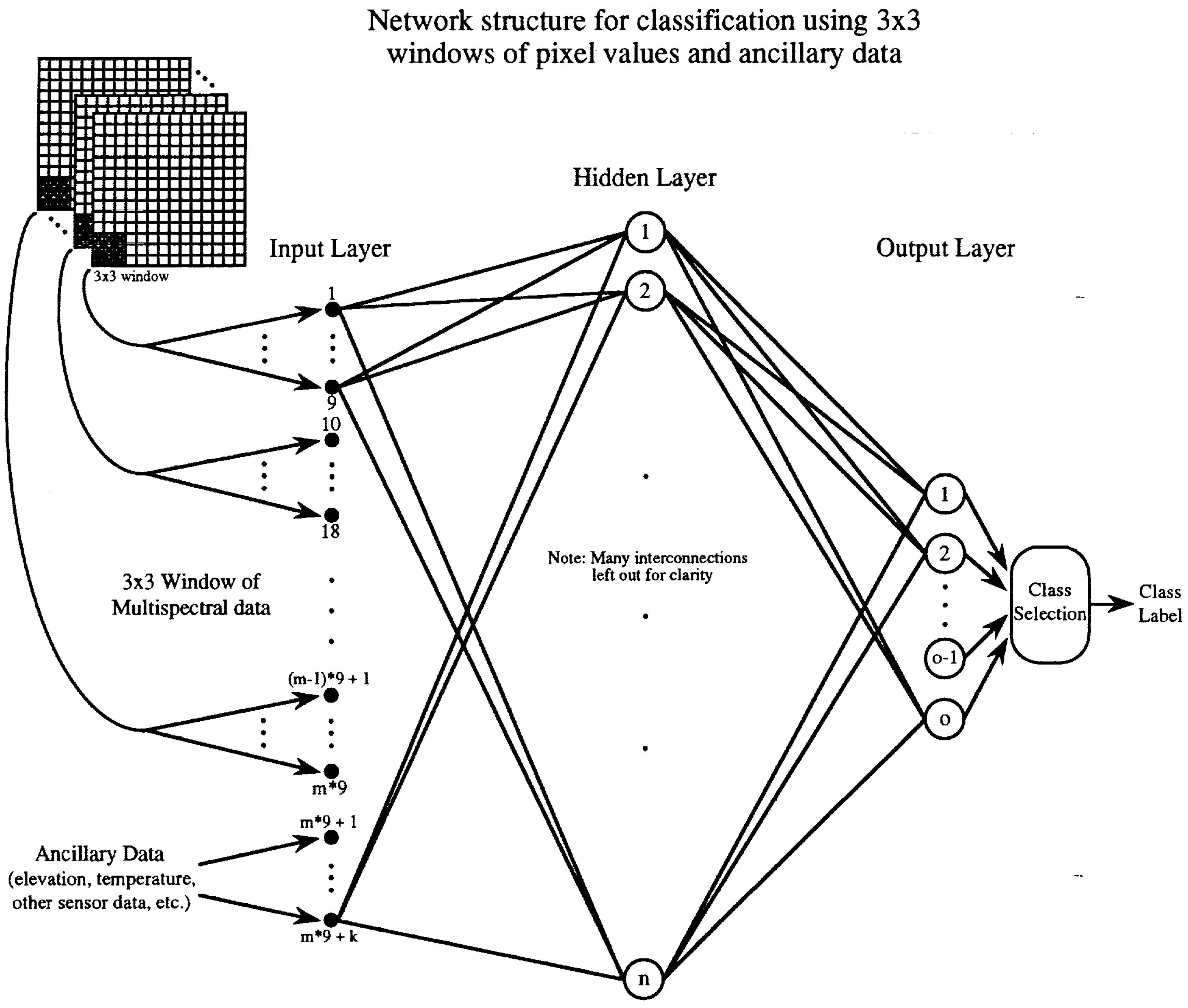

Network structure for classification using 3x3 windows of pixel values and ancillary data
Hidden Layer
Input Layer
Output Layer
3x3 window
1
9
10
18
3x3 Window of Multispectral data
(m-1)*9 + 1
m*9
m*9 + 1
m*9 + k
Ancillary Data (elevation, temperature, other sensor data, etc.)
Note: Many interconnections left out for clarity
1
2
n
1
2
o-1
o
Class Selection
Class Label

Figure 6: When using a window of multispectral data the number of interconnections and required nodes is greatly increased. In this case a 3x3 window is used in *m* bands of image data. In addition, *k* ancillary data parameters are fed in for each window. As in Figure 5, each output is used to represent *o* ground cover classes.

network should be avoided, however, as this can cause a substantial increase in training time with little gain in classification accuracy.

Encoding

Since, in the standard backpropagation neural network model, the inputs are connected to all hidden layer nodes, the order of the input data is not important. Beyond maintaining consistency of input data from one training sample to the next, the placement of the various nodes (i.e., starting with band 1 first or reading a 3x3 window from left to right and top to bottom, etc.) is irrelevant. However, the format of the number presented to the network is crucial. Most neural network algorithms are designed to deal with data ranging from 0 to 1. Although this is not a mathematical requirement, it avoids the use of a scale or shift factor every time the sigmoid activation function (Figure 3) is evaluated, thus achieving minimal floating point computations. Thus, a simple way to prepare the data for the network is to scale the value of each band of a pixel to this range, and present each scaled value to a separate input node. This encoding technique has been implemented in most of the papers surveyed.

Another encoding possibility is binary representation of the pixel value (Heermann and Khazenie, 1992; Benediktsson *et al.*, 1990b). Thus, in the case of 8-bit band values, 8 inputs are required. This would seem to be a poor choice because it would take 32 inputs to represent a 4 band pixel using this method and only 4 inputs using the simple scaling procedure. The justification for binary representation is that the network cannot sense the small differences in similar pixel values when they are scaled to such a small range (Bischof *et al.*, 1992). The arguments against this method, in addition to increased training time per cycle, are the fact that the bit patterns change dramatically at times with small changes in pixel value (e.g., from 127 to 128) and the fact that lower bits in the pixel are often noisy. Adjacent data values are assumed to be likely to belong to the same information class and therefore should have similar codes (Benediktsson *et al.*, 1990a). Alternatives to using a standard binary representation are to use gray coding (Benediktsson *et al.*, 1990a) or coarse coding (Bischof *et al.*, 1992). Gray coding is a modified binary encoding technique that produces numbers guaranteed to change by only one bit from one number to the next. Coarse coding can be viewed as a form of interpolation. An arbitrary number of input nodes is used to represent the desired input data range. These nodes are distributed uniformly in this interval. A Gaussian response function is then used to smear the input data number over this range of values to produce floating point values for each of the input nodes. According to Bischof *et al.* (1992), the advantage of coarse coding over gray coding is that in general, continuous codings are preferable to discrete codings.

2.2 Network output

Encoding

Once the input layer structure and data format have been determined a similar process must be undertaken with the output layer. The most straightforward way of encoding the output classes is to use one output node per ground cover class. Generally, the desired output values during the neural network training phase consist of low values (e.g., 0.1) for outputs that do not correspond to the pixel's assigned class and a high value (e.g., 0.9) for the output that does correspond to the pixel's assigned class. This method was used by almost all of the papers surveyed.

Benediktsson *et al.* (1990a) discuss two additional output encoding methods. The first is binary encoding. In this method only $\log_2 M$ output nodes are required to represent M classes, resulting in a decrease in training time per cycle over the one output per class case. However, as the authors note, the use of more than $\log_2 M$ output nodes can lead to fewer training cycles to achieve the same error, since the Hamming distance[3] of the output representations for the classes can be larger. Their second method is called temperature coding. As in the one output per class method, the number of outputs is equal to the number of classes, but the code for class n consists of a 1 value for the first n outputs and a -1 value for the remaining outputs. This results in a larger Hamming distance for the output representations of the classes than in the cases of both binary and one output per class encoding. Heermann and Khazenie (1992) attempted some classifications with binary class encoding for a reason other than decreased time per training cycle. Their motivation for this was to force the network to classify all the pixels

[3] The Hamming distance is the number of instances in which the corresponding entries of two binary vectors do not agree.

and disallow unknown pixel results. The mapping in this case was inferior to that of the one output per class technique. Civco (1991) took output node reduction even further than binary encoding by using a single output node. Presumably, the continuous range of the output value of that single node was partitioned into regions corresponding to the various classes. Use of a single output node would lead to an even more dramatic decrease in training time per cycle than that of binary encoding. However, the problems with binary encoding mentioned by Benediktsson *et al.* (1990a) are more acute in this case. The processing ability of one output node is limited. A single output node would require finely tuned weights to correctly partition the data space into the final classes all by itself. Convergence might even be impossible, making an increase in the number of hidden layer nodes necessary.

Extraction of classes

The next issue concerning the output data representation is how to interpret the continuous range of the output values achieved during the classification phase. Although the outputs are generally trained to be discrete values, there are two reasons why a continuous range from the low value to the high value will be seen during classification. First, the network is rarely trained to zero error on the training data. Thus, the outputs, even for the training data, will not exactly match the desired discrete values. More importantly, it is presumed that the data fed in for classification will be more diverse than the training data. Regions that do not fit easily into any training category will produce different sets of output values than those expected for the given classes.

For the rest of this section we will assume that the outputs were designed to represent one class apiece and were trained to have "high" values for their given class. The simplest way to assign a class to the input data is to choose the class of the output node with the highest value (e.g., Cromp, 1991; Mulder and Spreeuwers, 1991; Benediktsson *et al.*, 1990b). Key *et al.* (1990; 1989) modified this scheme with a threshold. The pixel was termed unclassified if all of the outputs (which ranged from 0 to 1) were less than 0.4; otherwise it was given the class of the highest valued output node. Another possibility is to consider the magnitude of all of the output values as a metric of some sort in determining a final class assignment.

There are two ways to interpret the set of continuous output values for the refinement of the classification. The first is to interpret them as classification confidence data. Higher values for an output would imply a higher confidence that the pixel belongs to the corresponding class. Short (1991) designed a system using neural networks as the first step in a real-time classification system for use with Earth Observing System (EOS) data. A neural network appropriate for the sensor type and climate being analyzed would do an initial classification and the outputs would be sent to an expert system for final classification. According to Short, the output of the neural network describes the confidence and ranking of class membership. Kiang (1992) discusses a method with which to convert the output values to *a posteriori* probabilities that can then be used improve the classification accuracy. Bischof *et al.* (1992) used a second neural network to smooth the original classification data. The input to this auxiliary network is a square window of the standard threshold classified image. For each pixel of the window, the classification result along with confidence information generated during classification are input to the net. The desired output classes are the same as for the original classification. With this method, the degree of smoothing depends on the confidence of the classification at each pixel.

The second way to view the continuous range of output values is as a measure of class mixing. McClellan *et al.* (1989) in a simple land and water classification used a 3 output network to represent the 2 classes. The continuous output data was scaled and viewed as an RGB image. The third output had been trained to 0.1 for all training data and was only included for the purposes of the RGB display. They discovered that in shoreline regions the first two outputs were nearly equal and in between the high and low training values. This was interpreted as the neural network clustering these areas into a separate mixed class. The nonlinearity of the network caused a clustering of this data around one point instead of stretching the outputs all along the range of the high and low training values.

The meaning of the continuous valued outputs cannot be determined for sure without comparisons with extensive ground truth data. However, there is an inherent fuzzy logic property to the neural network. Without any additional processing, a neural network with a one output node per class encoding can provide additional information beyond that of a "hard" classifier. Key *et al.* (1989) claim such a neural network classifier has both the confidence measurement and mixed class detection abilities. Using this interpretation, each output can be thought of as a membership value to a particular class. When a sigmoid activation function, which is linear through much of its output range, is used, these membership values are linear. Thus, a pixel can have a very high value in one node and low values in the others denoting a strong likelihood of belonging to that one class. If it has two high outputs then a mix of two classes is indicated. If one output is only slightly higher than the rest, then one class with a low confidence measure is noted. And if all outputs are low the network cannot fit the pixel to any of the trained classes or the pixel is a mixture of all classes.

Heermann and Khazenie (1992) exhibit the class mixing concept in a table showing the neural network classifier results. Along with the classification percentages for each of their five trained classes, they show the percentages for some two class mixes that were obtained. With real image data, class mixing is a common occurrence. Thus, the ability of neural networks to detect class combinations enriches the classification and brings it beyond the set of classes with which the network is trained. This ability should be especially apparent when texture information, such as a window of pixels is incorporated into the network training. Parametric classifiers can be used to detect mixed classes fairly easily when used on pixel data. A mixed pixel can be simply one with a value in between the means of two classes, for instance. But when texture or ancillary data is used these measures become more complex because the distributions will often differ greatly from that of the multispectral data. The neural network, on the other hand, will provide fuzzy output values for any type of input data.

2.3 Network Architecture

While the structure of the first and last layers of the neural network are controlled by external factors, the number of hidden layers and their size must be determined experimentally. Table 1 shows the data encoding techniques and network structures used by some of the researchers who discussed the topic of network architecture. The degree to which the theoretical capabilities of the structures shown in Figure 4 can be utilized depends on the number of hidden layer nodes. In the case of a single hidden

Table 1: Summary of some of the data coding techniques and network structures.

Authors:	Imagery:	Input Data Coding:	Output Data Coding:	Network Structure:
Benediktsson et al. (1990a)	MSS, elevation, slope, aspect data	Gray coding	Temperature and binary coding	56-32-10 and 56-32-4
Benediktsson et al. (1990b)	60 bands of Simulated HIRIS	Binary coding, 12 bits per band	3 outputs, one per class	240-15-3, 480-15-3, and 720-20-3
Bischof et al. (1992)	7 TM bands	Coarse coding, requiring 13 inputs per band. Also with 5x5, 7x7 window in band 5	4 outputs, one per class	91-5-4, 116-8-4, and 140-8-4
Dreyer (1993)	3 SPOT bands, texture calculations	Individual pixel values	9 outputs, one per class	3-13-12-9, 6-8-8-9, and 42-7-7-9
Heerman, Khazenie (1992)	3 TM bands	Binary data, 8 bits per band	5 outputs, one per class	24-24-5
Hepner et al. (1990)	4 TM bands	3x3 window of pixel values in each band	4 outputs, one per class	36-10-4
Kannellopoulos et al. (1991)	2 date SPOT	Individual pixel values	20 outputs, one per class	6-18-54-20
Key et al. (1990)	Merged AVHRR and SMMR	Individual pixel values	12 outputs, one per class	7-10-12
Li, Si (1992)	10 band airborne spectrometer	Individual pixel values. Input patterns are normalized first	3 outputs for 5 classes, coding unspecified	10-7-3
Wilkinson et al. (1992)	2 dates of 3 SPOT bands	Individual pixel values	7 outputs, one per class	3-15-7 and 6-21-7

layer, for instance, the output node that determines a given class serves as an AND operator on the half-plane decision boundaries produced by the hidden layer (Lippman, 1987). The actual decision regions will be convex polygons or unbounded convex regions with a number of sides equal to or less than the number of hidden layer nodes. Thus, the more hidden layer nodes that are used, the more flexibility the network has to carve up the input data into an appropriate decision space. If the network is allowed too much flexibility it can overtrain. In this case the final decision region is too specific to the training data and not applicable to the rest of the image. The optimal number of hidden layer nodes depends on the problem at hand, and it is necessary to determine this number by experimentation. If the training error does not decrease to an acceptable level, then the number of nodes should be increased. If the error becomes very small but the resulting classification is poor, then perhaps there are too many hidden layer nodes. Unfortunately, there are no specific theoretical guidelines for the size of the hidden layers.

Three layer networks

Generally for classification of multispectral imagery a three layer (single hidden layer) fully interconnected network is sufficient and is the most common implementation seen in the literature. A four layer net is often unnecessary since the class distributions do not usually consist of disconnected or other non-convex regions. In many of the surveyed papers a trial and error method has been used to determine the number of hidden layer nodes that result in the best classification. The result turned out to be that, in general, the number of hidden layer nodes used has been proportional to the number of output nodes (i.e., number of classes) and relatively independent of the number and format of the inputs. This makes sense since the output layer determines class by combining the hyperplane decision boundaries of the hidden layer. Thus a more complex classification using a larger number of classes will require more decision boundaries for the output nodes to choose from.

A few of the researchers listed in Table 1 indicated that they chose the number of hidden layer nodes by experimentation. Benediktsson *et al.* (1990b), with 20 bands represented as binary inputs used a structure that consisted of 240 inputs, 15 hidden layer nodes, and 3 output nodes. In the other paper by Benediktsson *et al.* (1990a), they determined that their 56 input, 10 output network required 32 hidden layer nodes. Bischof *et al.* (1992), using 13 input nodes per band (91 total inputs) and 4 output nodes, determined 5 hidden layer nodes to be optimal after trial runs with from 2 to 15 nodes. The classification performance did not change much with the hidden layer nodes ranging from 3 to 15. They attribute this to the large number of training samples used -- the performance did not decrease with more nodes, as they expected it to, because there were sufficient training samples to train the excess weights in the larger hidden layer configurations. Hepner *et al.* (1990), with a 3x3 window of pixel data, found that 10 hidden layer nodes were optimal for their 36 input, 4 output network. Fewer hidden layer nodes resulted in insufficient partitioning of the input space. More hidden layer nodes resulted in the training becoming stuck in local minima and thus not converging to a global minimum error.

Figure 7 shows plots of the number of hidden layer nodes as a function of the number of input layer nodes, input features, and classes for the three layer neural networks used in the papers in Table 1. The input data has an effect on the optimal number of hidden layer nodes in that the inherent separability of the data will help determine how many hyperplanes are required to separate the classes at the output layer. However, this separability is to a certain degree independent of the data representation and thus the number of inputs to the network. The classes of a three band image, for instance, will have the same inherent separability whether each band is represented in the network by a single scaled input or by a full 8-bit binary representation. As expected, the first plot of Figure 7 shows a definite lack of correlation between the number of input nodes and the number of hidden layer nodes.

To account for the differences in input structure and encoding techniques, the number of hidden layer nodes is also plotted in Figure 7 as a function of the number of features, defined as the number of multispectral bands and ancillary data sources used in the classification. Thus, the effects of using windows of input data and binary encoding techniques, for instance, are ignored, so that we can examine the relationship between the number of hidden nodes and the dimensionality of the input data. While the distribution of the second plot is less scattered than the first, it by no means indicates a correlation between number of hidden layer nodes and number of

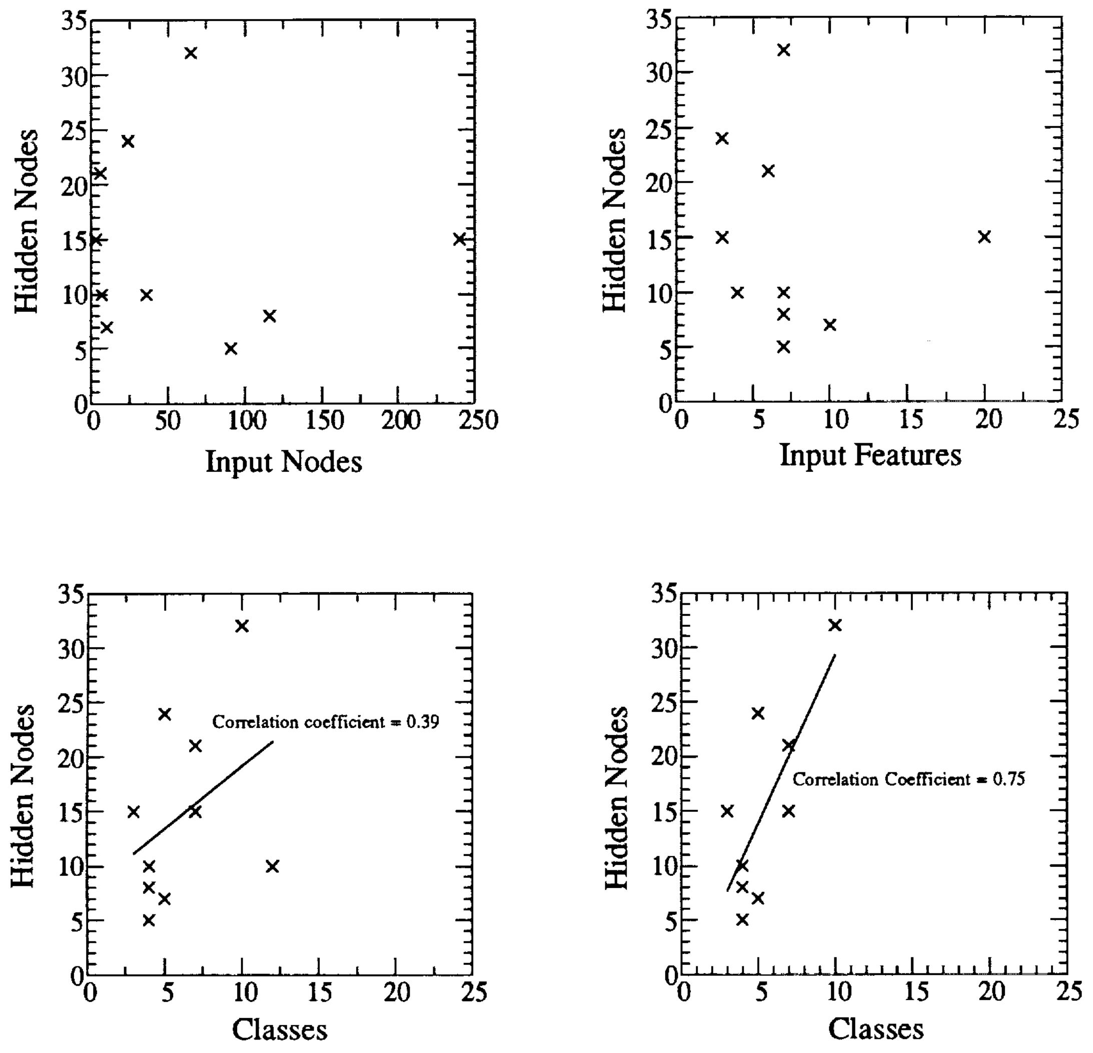


Figure 7: Hidden layer node size vs. number of input nodes, input features, and classes for the three layer neural networks used in the papers in Table 1. There is no correlation in this data between input nodes or features and hidden layer size. The plots of hidden layer nodes as a function of number of classes are shown with a linear fit. The lower right plot, showing a high degree of correlation between the hidden layer size and number of classes, is the result of deleting one atypical point from the data.

input features. The third plot, however, which shows the number of hidden layer nodes plotted as a function of the number of classes, shows a small degree of correlation between these two quantities. The correlation coefficient of a linear fit is 0.39, which can be increased to a respectable 0.75 if one stray point is removed. This point is from a cloud and surface classification using AVHRR and SMMR from polar regions (Key *et al.*, 1989) and can be considered atypical. The slope of the resulting linear fit is between 2 and 3, indicating that the number of hidden layer nodes used was 2 to 3 times more than the number of classes.

While this result is based on a small sampling, it is has some important consequences. It demonstrates that such analysis is possible to help develop a metric for one of the network's unknown quantities, the hidden layer size. It should be noted that not all of the network structures represented in the plot were optimized with respect to the number of hidden layer nodes. If this had been done, the correlation might be even stronger. Also, the data sources and classification complexities varied a great deal, so this small set was a decent representation of the range of problems encountered. Although the result does not precisely define the number of hidden layer nodes based on the number of classes, it does give a starting point for the hidden layer size with which to begin network experimentation. This is important because it helps to make an often confusing aspect of the neural network procedure more transparent.

Alternative numbers of layers

Some researchers have also looked into the use of two and four layer networks. Benediktsson *et al.* (1990b) used a two layer network (which they called a conjugate-gradient linear classifier) in the classification of high dimensional data. They found that the linear classifier obtained classification results that were similar to that of the standard three layer network. In their other paper (1990a) they comment that the use of a four layer network did not improve classification accuracy over the three layer case. Civco (1991) also obtained similar performance with the three layer (6-15-1) and four layer (6-6-15-1) network topologies. Kanellopoulos *et al.* (1991) used a four layer network with a 6-18-54-20 configuration. As a starting point for determining the number of hidden layer nodes they assumed that, for an N-dimensional feature space, the first hidden layer requires at least 2N nodes to define hyper-regions that form the basis of a classification (also Pao, 1989). Choosing the number of nodes in the second hidden layer is more complex. Their assumption was that the second hidden layer requires more nodes as the diversity of regions belonging to one class increases. In their initial experimentation, the addition of nodes in each hidden layer improved classification performance steadily until a point of minimal gain was encountered. Unfortunately they did not present the results of a three layer network for comparison. The classification discussed in this paper was one of the more complex (20 classes) of all the papers reviewed, and for this reason the arbitrary decision region capabilities of the four layer net may have been required to achieve the most accurate classification.

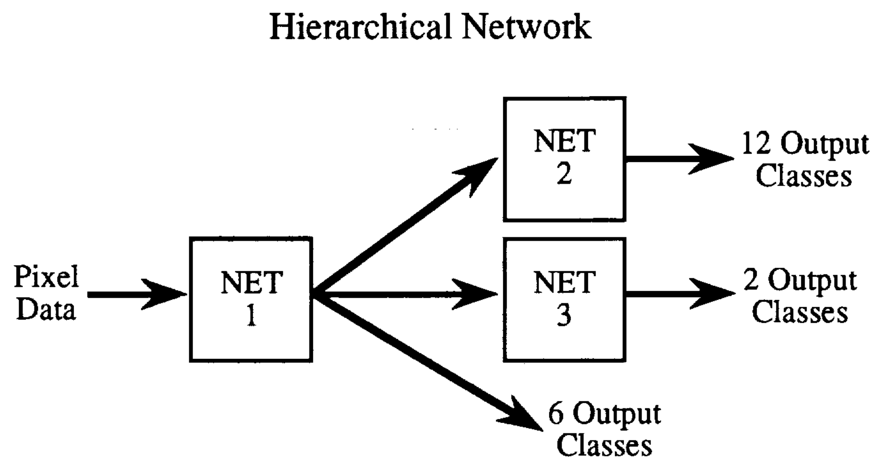
Dreyer (1993) tried both three and four layer networks and determined that the three layer net could not obtain as high a classification accuracy as the four layer one. An interesting aspect of Dreyer's work is the implementation of a hidden layer node reduction technique. He states that the hidden layer size may be optimized given a tradeoff between the ability of the network to generalize and the ability to train to minimal error. His solution was to simplify the network by removing insignificant hidden layer nodes based on a relevance measure after training, and then re-training the smaller configuration. The weights remaining after node deletion, which represent some degree of learning, are used at the start of the new training stage. This gives the weights a head start and speeds up the subsequent training phase. This method can be extended from the removal of hidden layer nodes to the removal of input features. If a node of the input layer has little significance, then its corresponding input feature can be left out of the analysis. Using his optimization technique, Dreyer reduced the number of hidden layer nodes by about half. A small improvement was noticed in the classification accuracy of the net after optimization. Even without this increase, however, the optimization boosted efficiency since the number of node interconnections was reduced by a factor of four, resulting in a similar reduction in computing time.

Alternative network architectures

A few researchers have addressed the use of alternative network structures rather than the standard fully interconnected two, three, and four layer nets. One modification is the use of hierarchical connections of smaller neural networks. Benediktsson *et al.* (1990b) implemented a parallel, self-organizing, hierarchical neural network (PSHNN). This structure consists of a series of independent self-organizing stages. At the output of a stage there is an error detection scheme. If an input vector is rejected according to this scheme, it is passed through a non-linear transformation and sent to the next stage. Each stage works with a transformed version of the input vector, and not with the output of the previous stage. Thus, during classification all the stages can be run in parallel since the stages do not depend on outputs from the previous stages. It is termed self-organizing because the number of stages is determined during training -- more complex problems requiring more stages. This structure was found to be both faster and more accurate than the conventional three layer backpropagation network.

Kanellopoulos *et al.* (1991) used a two level hierarchical network as one approach to their 20 class problem. Their justification for using a hierarchical net was the fact that remotely sensed data is often classified into hierarchical land cover schemes. The architecture was determined by calculating the separability of the classes using a metric called the Swain-Fu distance. Two nets in the second level were designed to take outputs from a single net in the first level (Figure 8). Additionally, some classes were determined directly out of the first lower level network. Very little classification improvement resulted, but the training time was halved over the four layer standard net implementation due to the simpler network structures used.

Liu and Xiao (1991) proposed an alternative neural network algorithm called blocked backpropagation to address the problems of slow error convergence and local minima, which they claim are commonly encountered when using standard backpropagation for remotely sensed image data sets. The hidden layer of this network is organized into blocks of units. The nodes in each block are linked to a single output cell (Figure 8). Thus, the hidden layer is not fully connected to the output layer. Liu and Xiao used four inputs, a hidden layer consisting of four blocks each of seven nodes, and four output nodes. They compared the results with those of a standard three layer



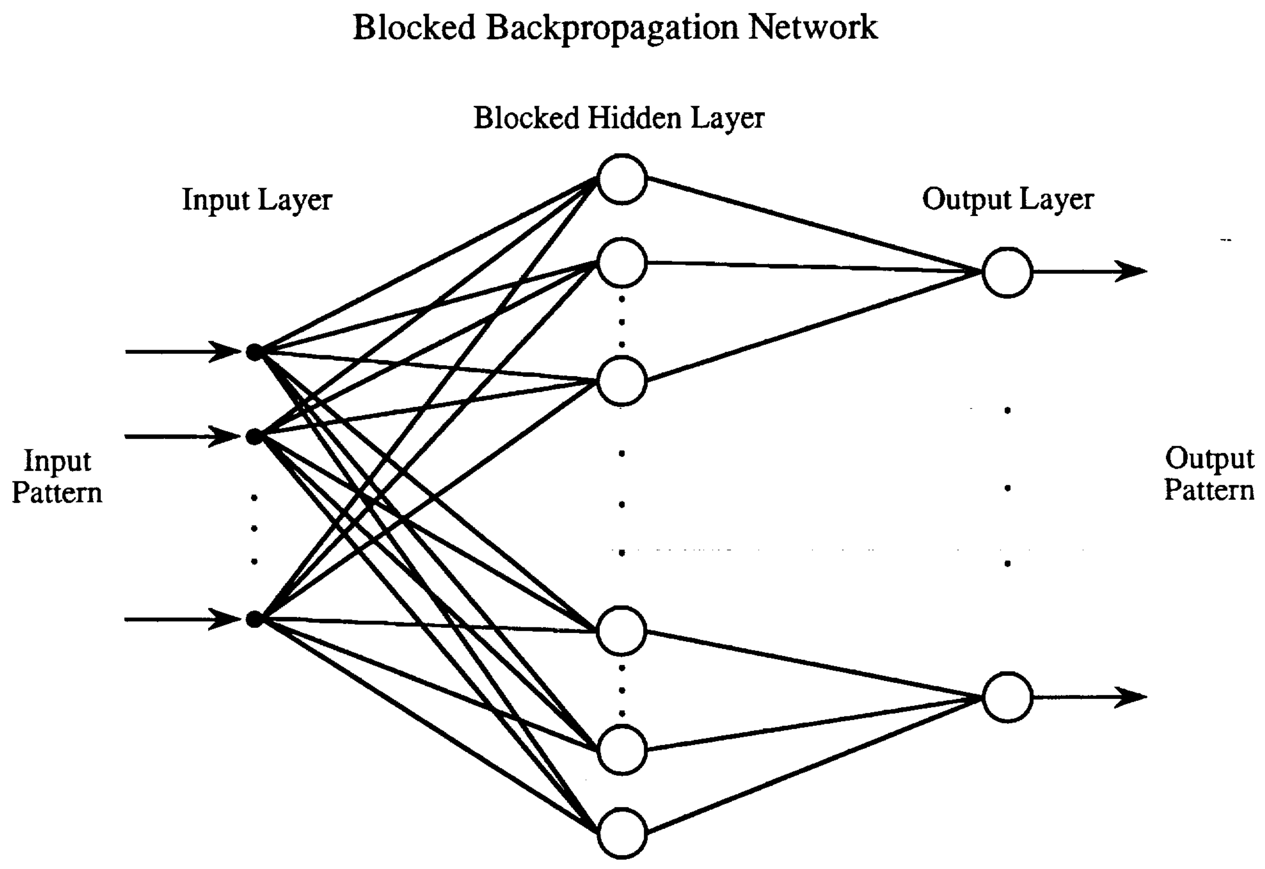


Figure 8: Alternative neural network structures. The hierarchical neural network used by Kanellopoulos *et al.* (1991) and the blocked backpropagation scheme of Liu and Xiao (1991).

backpropagation network with a 4-7-4 topology. The blocked backpropagation network was found to have faster and smoother convergence, and a higher classification accuracy than the standard net. It should be noted, however, that the standard net was only given seven hidden layer nodes for all four output nodes while the blocked back-propagation net was given a total of 28 hidden layer nodes, or seven for each output. Perhaps it would be fairer to compare the blocked backpropagation algorithm to a standard net with 28 hidden layer nodes, all fully interconnected to the output.

2.4 Training algorithms

Once the input data, class representation, and overall network topology have been determined, the following must be selected: training data, the network parameters such as learning rate and momentum, and the criterion for training termination. The first of these, training data selection, is a problem common to all supervised training algorithms. The data must be representative of the desired class with which it is tagged. In addition, these classes must have some separability in the input feature space for the classifier to be able to discern them. In the surveyed papers, the size of the training sets varied considerably. Civco (1991) used one training sample, the mean vector, per class. This approach was extreme compared to the rest. Hepner *et al.* (1990) used what they termed a "minimal training set" consisting of a 10x10 training site for each class. A few others used similar training set sizes (e.g., Liu and Xiao, 1991; Benediktsson *et al.*, 1990b). The largest training set used consisted of 22,000 patterns (Heermann and Khazenie, 1992). However, it was found that the accuracy of classification was not much different for this large set than for a much smaller set of 4200 patterns. According to the authors, the preparation of the training data is probably more important for accurate classification than the size of the training data.

Heermann and Khazenie (1992) devoted a section of their paper to the preparation of their training set. They refer to a "picking and packing" technique. The "picking" is accomplished by first employing an unsupervised clustering algorithm. Then small homogeneous regions are hand selected to represent the ground cover classes. The training data is then cleaned up ("packed") by removing any pixel values that are duplicated either within that class or within other classes. Eliminating duplicate pixels *within* a class avoids redundant calculations, and eliminating duplicate training values *between* classes ensures that the network will not be given contradictory training information. This method was probably chosen in this case because of the ambiguity of actual class membership for the training data. However, if the class values for the training data are well known (e.g., through extensive ground truth data), it may not be advisable to eliminate duplicate samples in different classes, because this might reduce the ability of the network to detect mixed classes. In addition, the elimination of duplicates within a class will cause those particular patterns to have less weight in the training of that class. If the data truly contains pixel values of a limited range then it might be good to leave the duplicates in the training data and thus allow the network to favor those common inputs more heavily. The best approach is probably determined entirely by the type of data and classes. The problem is essentially one of how representative the training data is of the actual desired classes.

For the backpropagation algorithm, learning rate and possibly momentum terms must be specified. As with the hidden layer size, there is currently no way of selecting these parameters except experimentation. A true gradient descent is obtained only if infinitesimal steps are taken (Rumelhart *et al.*, 1986). The most rapid learning occurs for the largest learning rate that does not lead to oscillation. The use of a momentum term allows the learning rate to be higher. Rumelhart *et al.* (1986), in some general backpropagation experiments, found that similar solutions can be obtained either with a zero momentum and small learning rate or with a higher momentum and higher learning rate. The second option will lead to faster training.

Kanellopoulos *et al.* (1991), however, found that the best results were obtained when the momentum term was eliminated and the learning rate was normalized to be proportional to the number of network weights. Heermann and Khazenie (1992) employed a similar technique but did not eliminate the momentum term. Their learning rate was set equal to $C_0 \cdot \frac{1}{P} \cdot \frac{1}{N}$, where N is the total number of nodes in the network, P is the total number of training patterns, and C_0 is a constant that they determined experimentally to be 10.

Heermann and Khazenie (1992) also explored the use of an adaptive learning rate, as discussed previously, to increase classification speed and avoid oscillation. Their algorithm involved increasing the learning rate if the last training iteration resulted in a decrease in the summed error over all training patterns. Conversely, the learning rate is decreased (but not allowed to go to zero) and the momentum term disabled if the error increases. Once the error begins to decrease again the momentum term is reinstated. This technique resulted in a 5 to 10 times speedup of the training process compared to the standard fixed learning rate procedure, with no loss in classification accuracy. The other potential benefit of an adaptive learning rate is in removing the responsibility of learning rate selection from the user and making it part of the transparent operation of the network.

A few papers discussed the final error to which the network was trained. Benediktsson *et al.* (1990a) used the value of the overall change in weights from one iteration to the next as a metric to terminate the training process. A more common method is to use a threshold on the total error between actual and desired outputs for all training patterns as a criterion. Bischof *et al.* (1992) minimized the "sum squared error" in 50 training cycles. 98.2% of the training data was correctly classified at this stage. Civco (1991) obtained a root mean square error of 0.18 (on a scale of 0 to 1) after 250,000 iterations of the class mean vectors. In the studies of Hepner *et al.* (1990), the amount of error between the network output and the desired output was reduced to 15% before training was terminated. This level was considered appropriate for an accurate classification. Kanellopoulos *et al.* (1991) found that it required 600 training cycles of half of the training data for 81% classification accuracy on the other half. An important issue to keep in mind when pursuing the minimization of error is the problem of generalization versus specialization. In many neural network problems the minimization of training error is inherently beneficial. However, when the application is a large scale classification of multispectral imagery it is prudent to consider the concept of overtraining. This can be a problem especially with four layer networks. Since these are capable of arbitrary decision regions, they might have a tendency to specialize too much on the training data and have little applicability to the rest of the image during classification.

Another disadvantage of the backpropagation algorithm, besides the requirement of specifying parameters such as the learning rate, is the lengthy time necessary for training. When used as a feed-forward classifier, a neural network is very fast (Heermann and Khazenie, 1992). However, the iterative process required to produce that feed-forward classifier is quite time and computation intensive. Although different network structures, training set sizes, and final error criteria were used in each case, the training times required in the surveyed papers illustrate the range and magnitude of the time involved. Some of the times required for reaching the error criterion using the standard backpropagation algorithm were 23 minutes on an IBM PC/AT (Key *et al.*, 1989), 3.1 CPU hours on a Vax 8600 (Hepner *et al.*, 1990), 4 hours on a Sparcstation 2 (Wilkinson *et al.*, 1992), 10 hours on a SUN workstation (Mulder and Spreeuwers, 1991), and 3 weeks on an HP 9000-385 (Heermann and Khazenie, 1992). The last case was improved to 3 days with the adaptive learning rate method discussed previously. An important observation by Heermann and Khazenie (1992) is that the network training does not proceed linearly. Most of the error convergence occurs during the very beginning part of training. The rate of improvement falls off dramatically as learning progresses. Thus, the convergence criterion is an important factor in determining training time. It may be that a slight loss in training data accuracy, and thus a great gain in training speed, does not affect the overall classification significantly.

Another problem in the training of a neural network arises from the initial assignment of random weights. Since this assignment is completely independent of the data, training time can be long and multiple training sessions can have different results -- i.e., the number of training cycles required to achieve the same error can vary considerably due to different initial weight configurations. Li and Si (1992) have addressed these problems with their technique of initializing the weights by a self-organizing method prior to backpropagation training. They used Kohonen's self-organizing algorithm (see Pao, 1989), which performs an unsupervised clustering procedure, to set the starting weights between the input layer and the hidden layer. Once the two dimensional grid of nodes in the Kohonen network have stabilized, the weights are scaled using a sigmoid function and used as the first set of weights in the standard three layer backpropagation neural network. The second set of weights (between the hidden and output layers) is randomized as before. This method resulted in a dramatic increase in training speed. To obtain a similar error rate the standard procedure of completely random weights required over 1000 iterations while the self-organizing method required only 26 iterations. This technique, in combination with an adaptive learning rate algorithm, could be an important step towards removing the uncertainty and inconsistency of using neural networks for the classification of multispectral imagery.

2.5 Comparisons to conventional classifiers

One of the best ways to prove the feasibility of the neural network method for classifying remotely sensed multispectral imagery is to compare it to conventional statistical classifiers. Some of the papers presented this comparison quantitatively. The consensus is that the two methods are very similar. A few researchers, however, found the standard methods to be better than the neural network method (e.g., Civco, 1991; Mulder and Spreeuwers, 1991; Benediktsson *et al.*, 1990b). Benediktsson *et al.*, who found the maximum-likelihood method to have an accuracy of 87.56% compared to 64.89% for the three layer backpropagation network, however state that their use of simulated Gaussian distributed data gives the maximum-likelihood method an unfair advantage. In this case the maximum-likelihood procedure is optimal and the neural network can only approximate its performance. Likewise, Civco, who used only the mean vectors of each class for training, states that this method does not exploit the inherent variability of the image data. The maximum-likelihood classifier was allowed use of this variability in calculating class statistics, but the neural network was not provided the same measure of randomness with which to create its internal class representations accurately.

Most of the authors, however, found that the neural network technique produced similar or superior classifications to those of the standard methods (e.g., Bischof *et al.*, 1992; Heermann and Khazenie, 1992; Kiang, 1992; Liu and Xiao, 1991; Medina and Vásquez, 1991; Short, 1991; Benediktsson *et al.*, 1990a; Hepner *et al.*, 1990; Key *et al.*, 1990; Ritter and Hepner, 1990; Key *et al.*, 1989). Bischof *et al.* (1992) found that the maximum-likelihood and three layer backpropagation networks achieved similar results. When the network was supplied with a window of pixel data, however, its accuracy increased another 4% relative to the ground truth. Hepner *et al.* (1990), who used a 3x3 window for network input, found in a qualitative comparison that the network classifier resulted in more homogeneous regions, sharper transitions, and more continuous features than the conventional classifier. Another advantage found in their studies is that smaller training sets were required for the neural network method (also Ritter and Hepner, 1990). This is possibly due to the fact that with a small training set a statistical classifier does not have enough data to describe the assumed distribution (Key *et al.*, 1990). Since the network assumes no particular distribution it does not require as much training data. This is important because it could save a significant amount of the labor involved in collecting ground truth for training sites.

Another advantage discovered for the neural network method is its greater generalization to both input data type and distribution. Statistical classifiers are the better choice if the data distribution is known (Bischof *et al.*, 1992; Benediktsson *et al.*, 1990a; Benediktsson *et al.*, 1990b). However, the neural network method has greater flexibility in that it can correctly classify data that contains pixels that differ significantly from those in the training regions (Key *et al.*, 1990; Key *et al.*, 1989). Additionally, the neural network method can be adapted to include multi-source or ancillary data simply by adding input lines. The network itself takes care of how much weight each source should have in the classification (Benediktsson *et al.*, 1990a). For the statistical classifier case, however, different techniques must be employed than those used on pure multispectral data. Also, with multi-source data the distributions tend to be less regular and thus the non-parametric neural network method might be a better choice (Benediktsson *et al.*, 1990a). Texture measures, such as windows of pixel data, are easier to implement with the neural network method as well. Some of the researchers point to this advantage and to its potential for increasing classification accuracy (e.g., Bischof *et al.*, 1992; Civco, 1991; Medina and Vásquez, 1991; Hepner *et al.*, 1990; Key *et al.*, 1990; McClellan *et al.*, 1989). The greater generalization capability of the neural network also leads to superior signature extension to other images not used in the training. Key *et al.* (1990) found that the network classified a much higher percentage of imagery collected on a second date than the maximum-likelihood method did. The network is more robust because it does not depend entirely on the class mean and covariance measurements which are likely to change from one date to another.

A disadvantage of the neural network method is the loss of interpretability due to departure from statistical theory (Kanellopoulos *et al.*, 1991; Key *et al.*, 1990; Key *et al.*, 1989). Since classes are no longer assumed to have well-defined distributions the mechanisms behind a classification are difficult to discern. Information about the decision regions and relative importance of the features are distributed over all the weight values. There are ways of viewing these weights, however. Bischof *et al.* (1992) used "weight visualization curves" to help explain the internal representation of the network and to aid in feature extraction. By examining the weights from the input to each hidden layer node, and then that hidden node's contribution to each output, they can relate some of the internal representations to the physical characteristics of the image bands and classes. They also site an example of feature reduction using Landsat Thematic Mapper data. The blue and green channels have similar weights and are thus highly correlated. One can be eliminated without reducing classification accuracy. Key *et al.* (1989) used a similar method to determine which channels are important for classifying certain classes and which channels are redundant. Image bands that are weakly weighted contribute very little to the classification and can be eliminated. Heavily weighted lines are traced from the input layer through the hidden layer to the output layer to determine which bands affect which classes. They state, however, that the relationships are not always clear and must be interpreted with care.

Perhaps the biggest disadvantage of the neural network method, as addressed earlier, is the length of training time. While this can be mitigated with adaptive learning rate techniques, it is probably the major reason why statistical methods will continue to be favored in the near future. However, there are two reasons why this might change -- increasing volume of image data in classification problems and recent advances in massively parallel computers. According to Key *et al.* (1989), as larger images are classified while the amount of training data remains the same, training time becomes a smaller proportion of overall classification time and the two methods will achieve similar speed. Thus, with the advent of EOS, and the need for rapid large scale ground cover classification, a neural network might become a more feasible alternative. Short (1991), in fact, proposes the use of multiple neural networks, trained for different sensors and climate types, to be used as the front end to an expert system for the real-time classification of EOS data.

An important facet of the application of the neural network to large image sets is that the compact nature of a neural network allows for easy distribution of the classification data without requiring the transfer of an entire image (Heermann and Khazenie, 1992). Once the network has been trained, the weights can be sent to distributed computing sites to perform the rapid, hard-wired network classification stage. To see if this is an advantage over the maximum-likelihood method we produced equations for the number of classification parameters that would have to be transferred in order to perform a classification without the training data. For a three layer network (which can perform classifications similar to maximum-likelihood), assuming one input per band and one output per class encoding, the equation is

$$\text{Classification parameters for Net} = 3 + \text{hidden layer nodes} \bullet (\text{bands} + \text{classes}). \quad (8)$$

The first term is the number of parameters needed to specify the number of nodes in each of the three layers; the second term is the number of weights between the nodes. For the maximum-likelihood method, the equation, derived from Swain (1978), is

$$\begin{aligned}\text{Classification parameters for ML} &= 2 + \text{classes} \bullet \text{bands} + \\ &\tfrac{1}{2} \bullet \text{classes} \bullet \left(\text{bands}^2 + \text{bands}\right).\end{aligned} \quad (9)$$

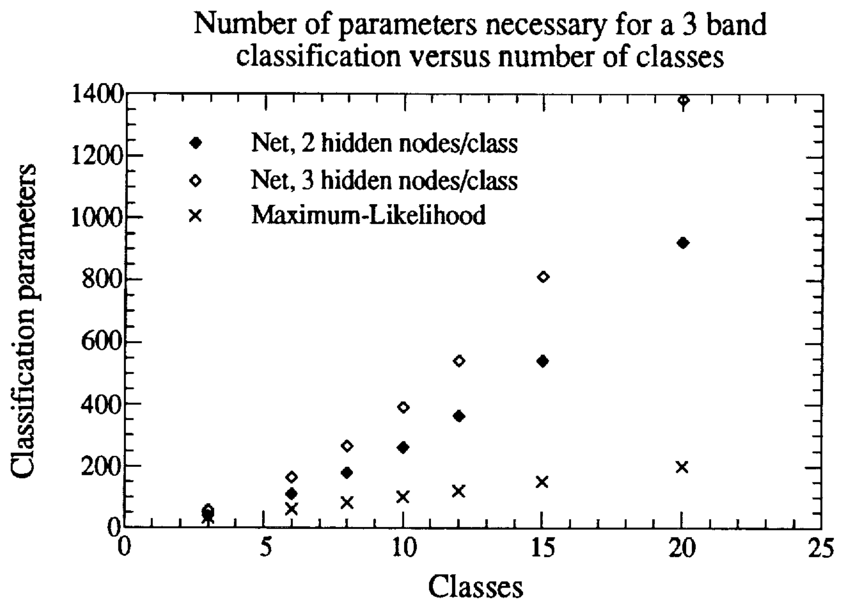
One parameter is the number of bands; one parameter is the number of classes; the second and third terms represent the number of parameters that are necessary for the evaluation of the quadratic during classification (see Swain, 1978). It can be seen from these equations that the number of parameters for the maximum likelihood method is quadratic with respect to the number of bands while the number of parameters for the neural network method will depend on the choice of hidden layer size.

Figures 9 and 10 contain plots of the number of classification parameters for the neural network and maximum-likelihood methods versus the number of classes in one case, and number of bands in the other case. Since little, if any, correlation was seen between the number of input and hidden layer nodes in a three layer network (see Figure 7), the number of hidden nodes was kept constant as the number of bands was varied. The empirical relationship between the number of hidden layer nodes and number of classes discussed previously, however, is reflected in the plots. In each plot the classification parameters are shown for networks containing 2 and 3 times as many hidden layer nodes as classes.

It can be seen from the plots of Figure 9 that, as the number of image bands is kept constant, the maximum-likelihood method is linear with respect to the number of classes, while the neural network is quadratic. Thus, as the number of classes is increased, the number of parameters required to reproduce the classification becomes much lower for maximum-likelihood than for the neural network methods. In the log plots of Figure 10 it can be seen that, as the number of classes is kept constant and the number of image bands is increased, however, the relationship is reversed. For a 200 band classification (e.g., AVIRIS) the network representation is 2 orders of magnitude smaller than that of maximum-likelihood. Since, in both methods, the majority of the parameters are used as elements of matrix multiplications, the number of calculations required, and thus the time for classification, will be proportional to the number of parameters. Thus, in terms of both size of representation and classification speed, the maximum-likelihood method is better for classifying many classes (i.e., more than 5) with small numbers of bands (i.e., less than 8). Based on the assumption of 2 to 3 times as many hidden nodes as classes and no correlation between the number of hidden nodes and number of bands, the neural network method is better for all other cases.

The second possible impetus for the use of neural networks for ground cover classification is the proliferation of massively parallel computers. Dedicated neural network hardware would certainly result in a great speedup of training time (Heermann and Khazenie, 1992). But also, the use of general purpose parallel computers should help. Although the parallel implementation of an inherently parallel structure is not necessary for the successful use of neural networks for multispectral image classification, it has been suggested as a future step in some of the reviewed papers (e.g., Heermann and Khazenie, 1992; Kanellopoulos *et al.*, 1991; Short, 1991; McClellan *et al.*, 1989).

There is more than one way to parallelize neural network algorithms. The most obvious is to place each node of the network at a node of the parallel



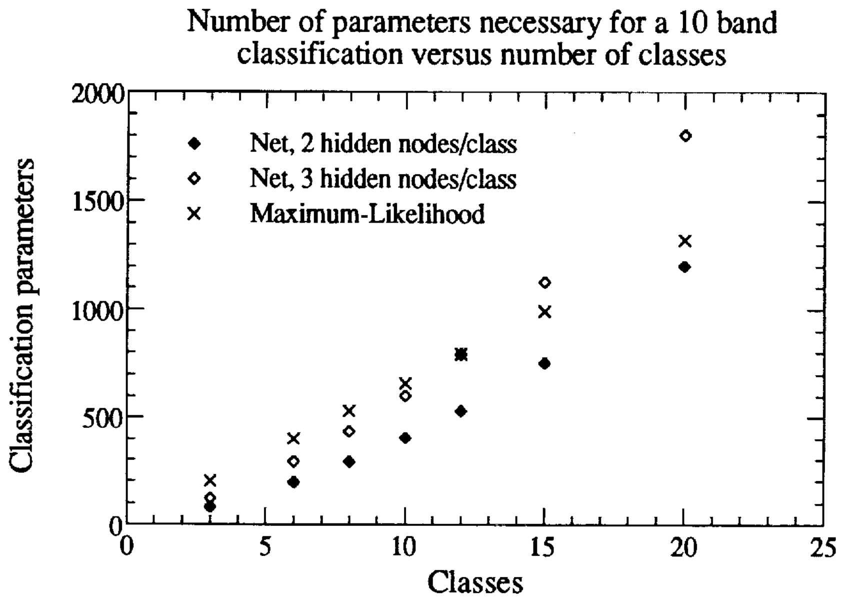
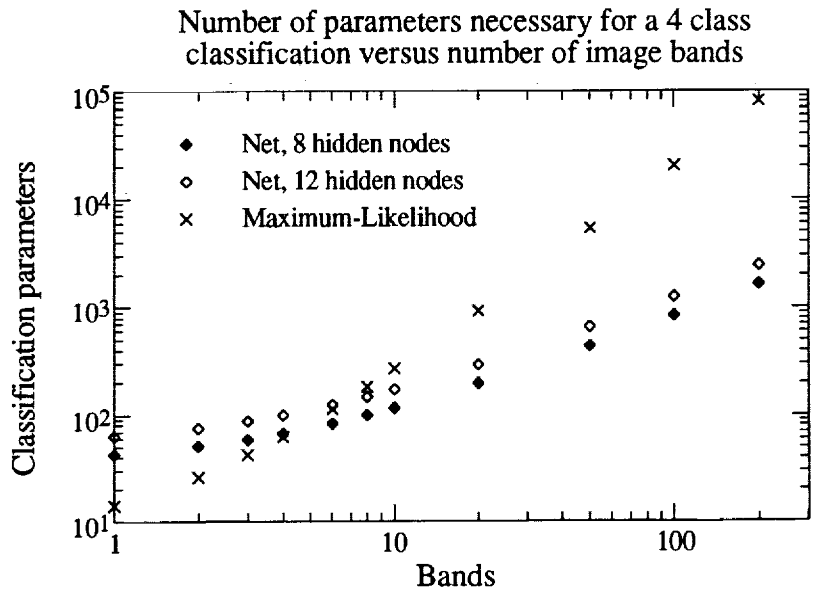


Figure 9: The number of parameters required to perform 3 and 10 band classifications without the original training data for both the neural network and maximum-likelihood methods are plotted as a function of the number of classes. For the network method it is assumed that one input node per band and one output node per class are used. The network is also assumed to have the empirically-derived relationship of 2 to 3 times as many hidden nodes as output nodes (Figure 7). This results in the network plots being quadratic with respect to the number of classes, while the maximum-likelihood plots are linear.



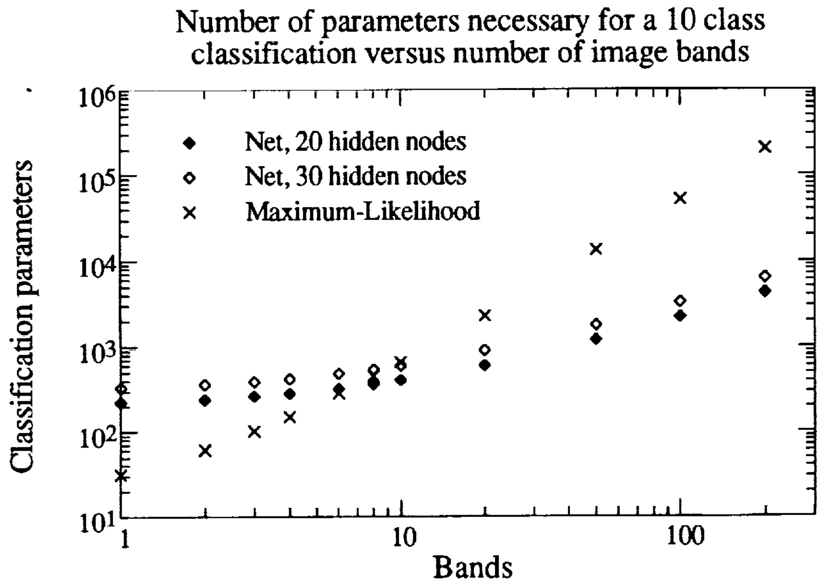


Figure 10: The number of parameters required to perform 4 and 10 class classifications without the original training data for both the neural network and maximum-likelihood methods are plotted on a log scale as a function of the number of bands (also on a log scale). Since the number of hidden layer nodes is assumed to be independent of the number of inputs (Figure 7), the network response is linear. The maximum-likelihood method response, however, is quadratic with respect to the number of bands.

computer. Another, perhaps easier, way is to implement the batch training method in parallel. During training, each pattern is presented to the network and the contributions of each training error are summed up. The total contribution is then used to adapt the weights before the next iteration. Since the training pattern presentations are independent, this procedure can be implemented in parallel.

3. SUMMARY AND CONCLUSIONS

The surveyed papers have established that the neural network approach is feasible for the classification of remotely sensed multispectral imagery. Similar performance to that of conventional classifiers, such as maximum-likelihood, has been achieved. However, further work is necessary before neural networks become a practicable alternative to conventional classifiers. While the method is easy to implement in software and adapt to different input data, and has shown a similar accuracy of classification as conventional classifiers, the slow and sometimes inconsistent learning phase, and the requirement of assigning obscure starting parameters such as learning rate, hidden layer size, and training convergence criterion make the approach less accessible to the casual user.

There are some current approaches for alleviating the problems of training time. The use of initial weight values derived from an unsupervised algorithm, such as the Kohonen self-organizing network, has been shown to both decrease training time and make the training process more consistent from one run to the next. An adaptive learning rate algorithm speeds up training by keeping the convergence steps at the largest possible size without causing oscillations. Implementations on parallel processing computers will also be important in keeping training time under control.

The other key to making the neural network approach competitive with conventional classifiers is to make the process more transparent to the user. Unfortunately, discussion of network parameters and experiments to develop metrics for these parameters have been sparse. The number of choices and parameters required to set up a training run can be overwhelming. The need for an empirically-derived learning rate can be eliminated by using an adaptive learning rate algorithm that adjusts as training progresses. Thus, the starting value of the rate is not as crucial to successful training. An examination of the surveyed papers indicates that there is some correlation between class number and another important experimentally-derived parameter, the hidden layer size. Further study of neural net classifications in which the hidden layer size is optimized with respect to classification error might lead to a better characterization of this relationship, which can then be used in subsequent classifications to determine a starting point for the network structure.

Once the technique of using neural networks for the classification of multispectral imagery has been developed into a useful tool, the challenge is to make use of the unique abilities of the network. Many of these capabilities have been investigated in a preliminary manner already. The ability of the network to assimilate many different types of data without the need for characterizing the distribution of each source can be exploited to expand the applications of the classifier. In addition, more varied input data structures are possible with the neural network than the single pixel and square window implementations already discussed. The inherent fuzzy nature of the output data has potential to provide class membership values and indicate class mixing with little subsequent processing. Some promise has been shown for a network's ability to generalize to other images with which it was not trained, even if the classes in these images have different first and second-order statistics. The ability of the four layer network to form disjointed decision regions allows it to be applied to classes with highly irregular distributions.

While all of these things are possible with conventional classifiers, the neural network approach provides them all in one algorithm. The general applicability of the neural network technique is both its strong point and its weak point, however. All the variables involved in the using a network, from the input structure and encoding, to the number of hidden layers and the choice of error convergence, while providing the great potential of the neural network, can easily intimidate someone interested in producing a ground cover classification. Thus, in order for the neural network to claim a niche as one of the tools of remote sensing it must be made into a faster, easier and more consistent method. Considering the potential benefits, this would be well worth the effort.

REFERENCES

Benediktsson, J. A., Swain, P. H., and Ersoy, O. K. (1990a), Neural Network Approaches Versus Statistical Methods in Classification of Multisource Remote Sensing Data, *IEEE Transactions on Geoscience and Remote Sensing* 28(4):540-551.

Benediktsson, J. A., Swain, P. H., Ersoy, O. K., and Hong, D. (1990b), Classification of Very High Dimensional Data Using Neural Networks, *Proceedings, 10th Annual International Geoscience and Remote Sensing Symposium*, College Park, Md., May, pp. 1269-1272.

Bischof, H., Schneider, W., and Pinz, A. J. (1992), Multispectral Classification of Landsat-Images Using Neural Networks, *IEEE Transactions on Geoscience and Remote Sensing* 30(3):482-490.

Caudill, Maureen (1988), Neural Networks Primer Part III, *AI Expert*, June, pp. 53-59.

Civco, Daniel L. (1991), Landsat TM Image Classification with an Artificial Neural Network, *Proceedings, ASPRS-ACSM Annual Meeting*, Baltimore, Md., Vol. 3, pp. 67-77.

Cromp, Robert F. (1991), Automated Extraction of Metadata From Remotely Sensed Satellite Imagery, *Proceedings, ASPRS-ACSM Annual Meeting*, Baltimore, Md., Vol. 3, pp. 111-120.

Dreyer, Peter (1993), Classification of Land Cover Using Optimized Neural Nets on SPOT Data, *Photogrammetric Engineering and Remote Sensing* 59(5):617-621.

Heermann, Philip D., and Khazenie, Nahid (1992), Classification of Multispectral Remote Sensing Data Using a Back-Propagation Neural Network, *IEEE Transactions on Geoscience and Remote Sensing* 30(1):81-88.

Hepner, George F., Logan, Thomas, Ritter, Niles, and Bryant, Nevin (1990), Artificial Neural Network Classification Using a Minimal Training Set: Comparison to Conventional Supervised Classification, *Photogrammetric Engineering and Remote Sensing* 56(4):469-473.

Kanellopoulos, I., Varfis, A., Wilkinson, G. G., and Mégier, J. (1991), Neural Network Classification of Multi-Date Satellite Imagery, *Proceedings, 11th Annual International Geoscience and Remote Sensing Symposium*, Espoo, Finland, June, pp. 2215-2218.

Key, J., Maslanik, J. A., and Schweiger, A. J. (1989), Classification of Merged AVHRR and SMMR Arctic Data with Neural Networks, *Photogrammetric Engineering and Remote Sensing* 55(9):1331-1338.

Key, J., Maslanik, J. A., and Schweiger, A. J. (1990), Neural Network vs. Maximum Likelihood Classifications of Spectral and Textural Features in Visible, Thermal, and Passive Microwave Data, *Proceedings, 10th Annual International Geoscience and Remote Sensing Symposium*, College Park, Md., May, pp. 1277-1280.

Kiang, Richard K. (1992), Classification of Remotely Sensed Data Using OCR-Inspired Neural Network Techniques, *Proceedings, 12th Annual International Geoscience and Remote Sensing Symposium*, Houston, Tx., May, pp. 1081-1083.

Li, Robert, and Si, Huaxiao (1992), Multi-spectral Image Classification Using Improved Backpropagation Neural Networks, *Proceedings, 12th Annual International Geoscience and Remote Sensing Symposium*, Houston, Tx., May, pp. 1078-1080.

Lippmann, Richard P. (1987), An Introduction to Computing with Neural Networks, *IEEE ASSP Magazine*, April, pp. 4-22.

Liu, Z. K., and Xiao, J. Y. (1991), Classification of Remotely-Sensed Image Data Using Artificial Neural Networks, *International Journal of Remote Sensing* 12(11):2433-2438.

McClellan, G. E., DeWitt, R. N., Hemmer, T. H., Matheson, L. N., and Moe, G. O. (1989), Multispectral Image-Processing With a Three-Layer Backpropagation Network, *Proceedings, 1989 International Joint Conference on Neural Networks*, Washington, D.C., Vol. 1, pp. 151-153.

Medina, Frances I., and Vásquez, Ramón (1991), Use of Simulated Neural Networks for Aerial Image Classification, *Proceedings, ASPRS-ACSM Annual Meeting*, Baltimore, Md., Vol. 3, pp. 268-274.

Mulder, N. J., and Spreeuwers, L. (1991), Neural Networks Applied To The Classification Of Remotely Sensed Data, *Proceedings, 11th Annual International Geoscience and Remote Sensing Symposium*, Espoo, Finland, June, pp. 2211-2213.

Pao, Yoh-Han (1989), *Adaptive Pattern Recognition and Neural Networks*, Addison-Wesley, Reading, Ma.

Richards, J. A. (1986), *Remote Sensing Digital Image Analysis: an Introduction*, Springer-Verlag, Berlin and Heidelberg.

Ritter, Niles D., and Hepner, George F. (1990), Application Of An Artificial Neural Network To Land-Cover Classification Of Thematic Mapper Imagery, *Computers & Geosciences* 16(6):873-880.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986), Learning Internal Representations by Error Propagation, In *Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Volume 1: Foundations* (D. E. Rumelhart, and J. L. McClelland, Eds.), The MIT Press, Cambridge, Ma., pp. 318-362.

Schowengerdt, Robert A. (1983), *Techniques for Image Processing and Classification in Remote Sensing*, Academic Press, Orlando.

Short, Nicholas (1991), A Real-Time Expert System and Neural Network for the Classification of Remotely Sensed Data, *Proceedings, ASPRS-ACSM Annual Meeting*, Baltimore, Md., Vol. 3, pp. 406-418.

Swain, Philip H. (1978), Fundamentals of Pattern Recognition in Remote Sensing, In *Remote Sensing: The Quantitative Approach* (P. H. Swain and S. M. Davis, Eds.), McGraw-Hill, New York, pp. 137-187.

Wilkinson, G. G., Kanellopoulos, I., Kontoes, C., and Mégier, J. (1992), A Comparison of Neural Network and Expert System Methods for Analysis of Remotely-Sensed Imagery, *Proceedings, 12th Annual International Geoscience and Remote Sensing Symposium*, Houston, Tx., May, pp. 62-64.