Working Paper

Artificial neural networks and statistical approaches to classifying remotely sensed data.

Rudolf T. Suurmond
Erik Bergkvist

WP-96-131
November 1996

IIASA International Institute for Applied Systems Analysis • A-2361 Laxenburg • Austria
Telephone: +43 2236 807 • Telefax: +43 2236 71313 • E-Mail: info@iiasa.ac.at

Artificial neural networks and statistical approaches to classifying remotely sensed data.

Rudolf T. Suurmond
Erik Bergkvist

WP-96-131
November 1996

Working Papers are interim reports on work of the International Institute for Applied Systems Analysis and have received only limited review. Views or opinions expressed herein do not necessarily represent those of the Institute, its National Member Organizations, or other organizations supporting the work.

IIASA International Institute for Applied Systems Analysis • A-2361 Laxenburg • Austria
Telephone: +43 2236 807 • Telefax: +43 2236 71313 • E-Mail: info@iiasa.ac.at

Contents

Preface

Remote sensing is a technique which provides enormous amounts of data for monitoring land use and land cover. In order to analyse the data statistical techniques can be used. Neural nets provide an interesting alternative.

The present Working Paper explores the capabilities of different types of neural nets in this respect and compares the results with the results of different advanced statistical methods.

The work is part of a cooperative study with the Department of Economic and Social Geography of the Vienna University of Economics and Business Administration. It was sponsored by the Austrian Ministry of Science.

Jaap Wessels

Department of Mathematics and Computing Science, Eindhoven University of Technology and IIASA.

Abstract

Substantial research has been done on automatic classification of remotely sensed data. Multispectral information about points on the earth is used to determine different types of land cover. One of the latest approaches to doing this type of research is the use of artificial neural networks. To train classifiers, we used a data set of pixels that were classified according to ground truth information. The pixels were taken from a Landsat-Thematic Mapper (TM) image of Vienna and its northern regions. The task was to classify the pixels into eight a priori defined categories of urban land use. Several different neural network paradigms were compared with more traditional statistical techniques. The self-organizing feature map performed the best in our study. It correctly classified 91% of the data in the test set. The best statistical method was the non parametric nearest neighbor method which correctly predicted 90% of the observations in the test set.

Acknowledgements

The research team gratefully acknowledges the help of professor Karl Kraus (Department of Photogrammetric Engineering and Remote Sensing, Vienna Technical University), for his assistance in supplying the remote sensing data. We would also like to acknowledge professor Manfred M. Fischer and Ms Petra Staufer (Institut für Wirtschafts- und Sozialgeographie, University of Vienna) for valuable ideas and suggestions that helped us improve the paper.

Different artificial neural network and statistical approaches in classifying remotely sensed data.

Rudolf T. Suurmond
Erik Bergkvist

1. Introduction

A number of satellites are continuously collecting data about the earth's surface. These data include spectral information that is, images of earth. Such images can be used to determine the type of land use or land cover in a particular region. In this study we explore the potential of several advanced techniques that facilitate the interpretation of such satellite images.

The data used in this study were taken from a Landsat-TM image of central Vienna and its northern regions. The image consisted of 270 × 360 (97 200) pixels each representing 30m × 30m areas. Six spectral bands were used: blue, green, red, near infrared, and two mid infrared bands. The thermal band with a resolution of only 120m was not used for classification. The spectral values were gray scaled in the 0–255 range.

In our study we interpret remotely sensed images by classifying them into eight urban land-use categories. Although clearly distinguishable from the ground data, land use is not easy to recognize from satellite images. Some categories are spectrally inhomogeneous, some are not spectrally separable. For each category, a group of pixels was chosen from the image. This is called the *one site condition.* The number of pixels in each category was chosen proportional to the total number of pixels of that category in the image. This resulted in a set of 2460 classified pixels. Two thirds of these pixels were used to estimate free parameters of the classifiers (learning), while the remaining one third were used to measure the accuracy of the classifiers (testing). The same learn and test sets were used in all classifiers discussed in this paper (with the exception of Sections 10 and 11).

The primary objective of the study was to explore the potential of neural networks in this classification task. But the results that are obtained from neural classifiers become considerably more interesting when they are compared with the results that can be obtained from other advanced methods. The best methodology can be determined only by applying different approaches to the same data and then compare the results.

Among the group of neural network classifiers we compared the performance of networks that use unsupervised learning with those that use supervised learning. Among the group of statistical classifiers, we compared parametric and nonparametric methods. Since artificial neural networks are essentially nonparametric classifiers, their performance is best assessed by comparing the corresponding statistical methods.

In this report we discuss the results we obtained from several different artificial neural networks that were used to solve this classification problem. First we discuss the general neural network approach. Next we investigate the self-organizing feature map, learning vector quantization, fuzzy ARTMAP, back-propagation, and radial basis function networks. Then, we report on several statistical methods to solve this and compare the results. Finally, we investigate the effect of modifications to the learn set on the learning vector quantization network.

Table 1: Land-use categories and numbers of training and testing pixels.

		Number of pixels	
Category	Description of the category	Learning	Testing
1	Mixed grass and arable farmland	167	83
2	Vineyards and areas with low vegetation cover	285	142
3	Asphalt and concrete surfaces	128	64
4	Woodland and public gardens with trees	402	200
5	Low density residential and industrial areas (suburban)	102	52
6	Densely built up residential areas (urban)	296	148
7	Water courses	153	77
8	Stagnant water bodies	107	54
	Total number of pixels	1640	820

2. Neural Network Classifiers

Although the internal structure and operation of the different neural classifiers discussed in this paper are quite diverse, it is possible to describe the general approach we used for training and testing them. In this section we will describe the software, and the accuracy measures used in the research and identify a few implementation issues.

The neural networks were implemented using the software package NeuralWorks Professional II/Plus. The exact algorithms used in this package are listed in the software manual [1]. In NeuralWorks, learning parameters are varied according to a so-called *learn schedule*. During the beginning of the learning process, we used parameter values that allow fast learning. The parameters are adjusted in later stages of the learning process to prevent oscillation and to fine-tune the weights. The learn count determines which column of the learn schedule is active. In Table 2, the learning rate is 0.06 for the first 12 300 learning iterations, 0.03 for the iterations between 12 301 and 24 600, and so on. Decreasing the learning rate after a number of learning iterations usually gives improves the results. In Table 2 the total number of learning iterations is 49200.

NeuralWorks is a versatile package that supports many different neural network paradigms. However, the facilities in this package for evaluating the performance of neural networks are relatively limited. It is not even possible to calculate the percentage of correct classified pixels in a set. Therefore, the evaluation was done by saving the output of the network and calculating the desired accuracy measures by means of a specially written C program.

The most important accuracy measure of a neural classifier is the average *classification rate*. The classification rate is the proportion of correctly classified pixels in a set of pixels. This value was averaged over five networks that were trained with different random seeds. A different random seed means that the initialization of the weights and the order in which the learning examples are presented are varied. Because we trained five nets, we were also to calculate the standard deviation and confidence intervals of the classification rates. These measures the stability of the network with different random initializations.

The second accuracy measure is the *classification matrix*. This matrix has eight rows and eight columns; each row corresponds to a ground truth category and each column to a category assigned by the classifier. The component in row *n* and column *m* gives the number of pixels of ground truth category *n* that were classified as category *m*. This accuracy measure shows which categories are confused by the classifier and is therefore also called confusion matrix.

All accuracy measures were calculated for both the train set and test set. The performance on the train set indicates how well the network is able to distinguish the pixels that were used to estimate the free parameters in the network. The performance on the test set, which is generally lower than on the train set, is the more interesting one. It is an indication of the classifiers ability to correctly classify pixels it has never seen during the training process.

The coding of the inputs and outputs can have major influence on the performance of a neural network The activation functions in the input layer are often very sensitive to the way in which the inputs are scaled. The best results are obtained when the inputs range between zero and one (and the weights initialized to random values in appropriate intervals). The inputs have different distributions. None of the inputs uses the full 0–255 range. Therefore, the inputs were scaled individually so that each scaled input used the full range [0.2,0.8].

The outputs were implemented with a one-of-*N* code. This means that our classification problem had eight outputs, one for each category. Each category was coded as a vector with one component equal to one and the remaining components equal to zero. This coding generally gives better results than coding the category as one numeric value.

3. Self-organizing Feature Maps

The self-organizing feature map (SOM), developed by Kohonen [2], is a neural network that creates a mapping of the input space onto a discrete, two-dimensional space while preserving the order of the input space. Thus, if two input vectors are close, then the images of these two input vectors will also be close. Each unit in the Kohonen layer represents an image value. For our purpose the most important aspect of the self-organizing feature map is that the mapping can also be regarded as a classification. In short, each unit in the Kohonen layer represents an image value of the mapping which represents a category.

The resulting classification is based not on the a priori classification provided in the learning examples, but only on the structure of the inputs. The mapping is such that each Kohonen unit is associated with an approximately equal number of input vectors and pixels that are close to one another in input space are associated with the same Kohonen unit (i.e. image value or category).

3.1 Kohonen learning and recall

The SOM network consists of two layers of units: the input layer and the Kohonen layer, which is in our application also the output layer. The Kohonen layer is twodimensional in the sense that its processing elements are arranged into rows and columns. When a learning example is fed to the network, the Euclidean distance to each Kohonen unit is calculated and the nearest unit is chosen to be the winner. The weights of a group of units around the winning unit are then moved in the direction of the learning example.

More formally, the input vector is denoted by $X = (x_1, ..., x_M)$. Since the input and Kohonen layers are fully connected, each Kohonen unit has *M* weight values that can be denoted by $W_i = (w_{i1}, w_{i2}, \mathrm{K}\ , w_{iM})$. The Euclidean distance D_i is calculated for each Kohonen unit as follows:

$$D_i = \left|X - W_i\right| = \sqrt{(x_1 - w_{i1})^2 + \cdots + (x_1 - w_{i1})^2}\ .$$

During recall the unit with the minimum D_i is declared the winner. During learning, a bias is added to D_i. The value of the bias depends on the frequency with which the unit has won in the past. It is calculated by

$$B_i = \gamma(N \cdot F_i - 1)\,,$$

where *N* is the number of units in the Kohonen layer, F_i is an estimation of the win frequency of unit *i,* and γ is the *conscience* parameter set by the user. The unit for which $D'_i = D_i + B_i$ is minimal is the winner.

The win frequency estimations are initialized to $F_i = \dfrac{1}{M}$ for all $i = 1, ..., N$. After each learning iteration, with winning unit *j*, they are updated to

$$F_{i\,\mathrm{new}} = F_{i\,\mathrm{old}} + \beta(\delta_{ij} - F_{i\,\mathrm{old}})\,,$$

where δ_{ij} is the Kronecker delta function and β the *frequency estimation* parameter set by the user.

During learning, the weights of the winning unit *j* and of a group of units called the *neighborhood* is updated according to the formula

$$W_{ij\,\mathrm{new}} = W_{ij\,\mathrm{old}} + \alpha(X_j - W_{ij\,\mathrm{old}})\,,$$

where α is the *learning rate*, which is a learning parameter set by the user.

The shape of the neighborhood can be either square or diamond. It consists of all units within a square or diamond around the winning unit.

Upon recall, the unit closest to the input vector with respect to Euclidean distance is chosen to be the winner and is the only Kohonen unit to have nonzero output.

After the SOM has been trained, it’s classification is analyzed and compared with the a priori classification provided by the learning examples. Since there are more Kohonen units than a priori categories, a post-processing phase was implemented that associates every Kohonen unit with the most appropriate a priori category. For each unit in the Kohonen layer, the group of learning examples that activates this unit is considered. We determined to which a priori category the majority of these learning examples belong. Then, when applying the network to unclassified data, the data were first presented to the SOM net, which selected a Kohonen unit. Finally, it is checked which a priori class had been associated with this unit and the unclassified vector is assigned to this category.

3.2 Experiments

Simulations were done for three different numbers of units in the Kohonen layer. The networks were trained for 49200 iterations each, according to the learn schedule given in Table 2. Experiments were also carried out with a constant learning rate of 0.06 for every learn count. The results from these can be seen in Table 3.

Table 2: Learn schedule for self-organising feature map.

Learn count	12300	24600	36900	49200
Learning rate	0.06000	0.03000	0.01500	0.0075
Frequency Est.	0.00100	0.00050	0.00025	0.00013
Conscience	1.50000	0.75000	0.37500	0.18750
Neighborhood width	7	5	3	1
Neighborhood shape	Square	Diamond	Square	–

Table 3: Performance of the SOM network with learn schedule given in Table 2. The 95% confidence intervals for the classification rates (lower bound, average, upper bound, respectively are listed in boldface).

Units			Train set			Test set		
Rows	Cols	Total	Classif. rate	Average	Std. dev.	Classif. rate	Average	Std. dev.
4	4	16.000	0.893	0.898	0.008	0.867	0.868	0.004
			0.889			0.876		
			0.897			0.863		
			0.899			0.867		
			0.911			0.868		
		C.I 95 %	**0.881**	**0.898**	**0.914**	**0.859**	**0.868**	**0.877**
10	10	100.000	0.922	0.922	0.001	0.888	0.896	0.011
			0.922			0.895		
			0.923			0.902		
			0.921			0.911		
			0.924			0.884		
		C.I 95 %	**0.920**	**0.922**	**0.925**	**0.874**	**0.896**	**0.918**
16	19	304.000	0.940	0.938	0.004	0.899	0.890	0.009
			0.935			0.878		
			0.933			0.893		
			0.938			0.896		
			0.941			0.884		
		C.I 95 %	**0.931**	**0.938**	**0.945**	**0.873**	**0.890**	**0.907**

Table 4. Performance of the SOM net with the learning rate 0.06. The 95% confidence intervals for the classification rates (lower bound, average, upper bound, respectively are listed in boldface).

		Train set			**Test set**	
	Classif. rate	**Average**	**Std. Dev.**	**Classif. rate**	**Average**	**Std. Dev.**
	0.930	**0.929**	**0.002**	0.911	**0.910**	**0.002**
	0.928			0.913		
	0.929			0.910		
	0.926			0.910		
	0.932			0.909		
C.I. 95%	**0.925**	**0.929**	**0.933**	**0.907**	**0.910**	**0.914**

The classification matrices for the second SOM network in Table 4, with 100 Kohonen units and a constant learning rate, are shown in Table 20 in the appendix. The whole data set consisting of 97 200 pixels were used with the SOM, LVQ and RBFN nets to create an image of Vienna (see Figures 9, 10 and 11 in the appendix). This was done to see if they show similar behaviour when it comes to generalization on a greater data set. Here it is apparent that the SOM net is better when it comes to separating group 6 from 7 (urban areas and water courses areas).

3.3 Discussion

The SOM network gives the best results when applied to many units in the Kohonen layer. This makes that the use of the SOM network requires considerably memory and computational power.

The performance of this type of network is remarkable because only part of the available information is used for learning. Indeed, the learning algorithm of the SOM uses only the input vectors and not the desired output vectors. In section 4 we will modify the SOM network to take into account the desired outputs during learning, namely, the learning vector quantization.

4. Learning Vector Quantization

A learning vector quantization (LVQ) network combines the advantages of a Kohonen layer and supervised learning. Unlike the self-organizing feature map, the Kohonen layer in an LVQ network does not preserve the order of the input space. Instead, each a priori category is assigned a fixed number of units in the Kohonen layer. During learning, the weights of each Kohonen unit move toward learning examples in its assigned category and move away from examples in other categories. A conscience mechanism similar to that used in the self-organizing feature map was implemented to prevent the same Kohonen unit from winning too frequently.

Like the self-organizing feature map, the learning vector quantization neural network paradigm was introduced by Kohonen [3].

4.1 Experiments

Of the network paradigms in our study, we explored the learning vector quantisation most thoroughly. First, we determined the optimal number of units in the Kohonen layer and the optimal number of learning iterations. Next we investigated the sensitivity with respect to other learning parameters.

The learn schedule used in most experiments is given in Table 5. The explanation of the different learning parameters as well as the exact algorithms used can be found in the software manual [1]. The first 75% of the learning iterations were LVQ1 iterations; the remaining iterations were LVQ2 iterations.

Table 5: Learn schedule for LVQ nets. The learn count is given as a fraction of the total number of learning iterations.

Learn count (fractional)	0.25	0.5	0.75	0.875	1
Attraction rate 1	0.06000	0.03000	0.01500	0.03000	0.01500
Attraction rate 2	0.06000	0.03000	0.01500		
Repulsion rate		0.03000	0.01500		
Conscience	1.50000	0.75000	0.37500		
Frequency estimation	0.00060	0.00030	0.00015		
LVQ2 width				0.20000	0.40000

4.1.1 Number of units and learning iterations

The performance was measured for all combinations of five different numbers of Kohonen units and five different numbers of learning iterations. To determine the preferable values of parameters, the average performance is important as well as the standard deviation. Therefore, both are presented in this paper. In this subsection we present the results of the experiments with different numbers of Kohonen units and different numbers of learning iterations in the various tables and figures.

The performance on the train set is as expected: increasing monotonically with both the number of units and learning iterations. The performance on the test set is more interesting. Almost all values are in the range between 85% and 90%. The best results have been obtained with either a small number of units or a large number of units. The difference between using 16 units or 304 units is only 0.2%, whereas the best values for network sizes in between is 1% to 2% less. Therefore, we continued with 16 units in the Kohonen layer.

The number of learning iterations behave remarkably in comparison with the number of Kohonen units. The networks between 56 and 200 units work best when trained for only 16 400 iterations, while the networks with 16 and 304 units require significantly more learning iterations.

Table 6: Average classification rates on the learn set.

	Learning iterations				
Units	16400	32800	49200	65600	82000
16	0.910	0.912	0.923	0.925	0.917
56	0.916	0.925	0.923	0.924	0.923
104	0.933	0.940	0.944	0.944	0.949
200	0.937	0.948	0.960	0.963	0.969
304	0.916	0.946	0.957	0.965	0.970

Table 7: Average classification rates on the test set.

	Learning iterations				
Units	16400	32800	49200	65600	82000
16	0.879	0.874	0.893	0.899	0.893
56	0.873	0.870	0.873	0.871	0.872
104	0.890	0.881	0.886	0.882	0.886
200	0.892	0.887	0.870	0.874	0.871
304	0.853	0.900	0.892	0.874	0.879

Table 8: Standard deviations of classification rates on the test set.

	Learning iterations				
Units	16400	32800	49200	65600	82000
16	0.010	0.008	0.011	0.004	0.015
56	0.006	0.004	0.005	0.005	0.005
104	0.009	0.007	0.004	0.007	0.006
200	0.007	0.003	0.006	0.003	0.006
304	0.022	0.009	0.008	0.008	0.007

Figure 1: Average classification rates on the learn set.

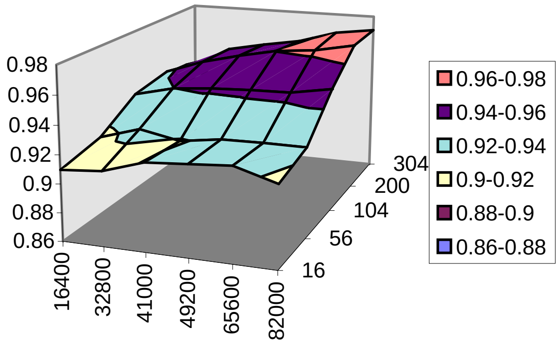


Figure 2: Average classification rates on the test set.

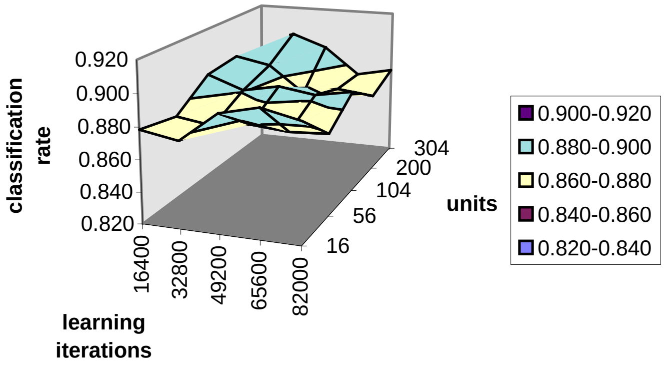
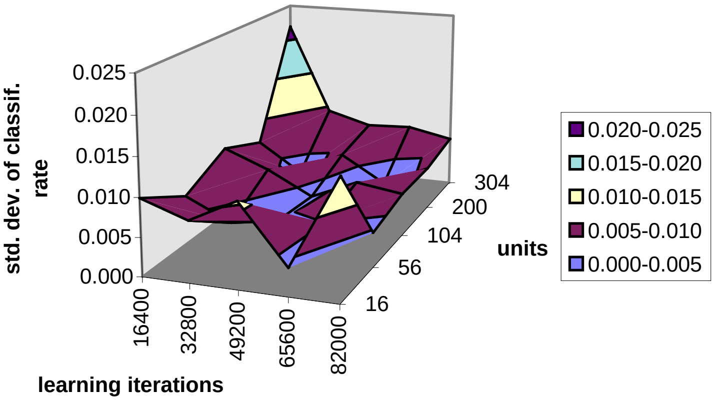


Figure 3: Standard deviations of classification rates on the test set.



4.1.2 Sensitivity to different learning parameters.

We analyzed the sensitivity of four different learning parameters on the small net with 16 hidden units and 65 600 learning iterations. One parameter was varied and the other parameters were kept at the values listed in Table 5. In Tables 9, 10, 11 and 12 list only the initial values for the parameters. We can complete the learn schedule for the nets by maintaining the same ratios within rows as used in Table 5. The classification matrices shown in Table 21 in the appendix have been obtained from a 16-unit network that was trained for 65

600 iterations. The learning parameters are shown in Table 5. Results from these experiments can be seen in Table 6, Table 7 and Table 8 with their graphical representation in Figure 1, Figure 2 and Figure 3 respectively

Tables 9, 10, 11 and 12 show stable behaviour with respect to most learning parameters, although each parameter has an interval that gives optimal results.

Table 9: Sensitivity of LVQ1 with respect to learning rate.

	Train set		Test set	
LVQ1 LR	Average	Std. dev.	Average	Std. dev.
0.01	0.919	0.002	0.889	0.009
0.06	0.925	0.004	0.899	0.006
0.08	0.915	0.015	0.886	0.020
0.10	0.924	0.002	0.901	0.006
0.20	0.919	0.011	0.887	0.013
0.40	0.919	0.008	0.885	0.009
0.80	0.921	0.007	0.883	0.010
1.00	0.915	0.009	0.881	0.010

Table 10: Sensitivity of LVQ1 with respect to conscience.

	Train set		Test set	
Consc.	Average	Std. dev.	Average	Std. dev.
0.00	0.910	0.002	0.843	0.006
0.50	0.912	0.009	0.876	0.008
0.75	0.922	0.007	0.898	0.015
1.00	0.925	0.003	0.902	0.004
1.50	0.925	0.004	0.899	0.006
2.00	0.919	0.005	0.892	0.010
3.00	0.924	0.005	0.900	0.004

Table 11: Sensitivity of LVQ2 with respect to learning rate.

	Train set		Test set	
LVQ2 LR	Average	Std. dev.	Average	Std. dev.
0.00	0.904	0.003	0.872	0.006
0.01	0.909	0.002	0.878	0.006
0.03	0.925	0.001	0.899	0.004
0.10	0.922	0.007	0.898	0.006
0.30	0.919	0.007	0.881	0.014
0.50	0.880	0.034	0.835	0.042

Table 12: Sensitivity of LVQ2 with respect to width.

	Train set		Test set	
LVQ2 width	Average	Std. dev.	Average	Std. dev.
0.01	0.902	0.005	0.865	0.010
0.05	0.921	0.002	0.886	0.001
0.10	0.920	0.008	0.886	0.005
0.20	0.925	0.001	0.899	0.004
0.30	0.918	0.008	0.897	0.006
0.40	0.921	0.005	0.895	0.011
0.50	0.923	0.003	0.900	0.004
0.60	0.910	0.025	0.890	0.021
0.80	0.910	0.020	0.883	0.019
1.00	0.913	0.012	0.888	0.016

5. Fuzzy ARTMAP

The fuzzy adaptive resonance theory MAP (FZARTMAP) comprises three main parts: two ART networks and a match tracking system (MTS) that connects the two. In contrast to a regular ART1 network, the FZARTMAP can handle continuous input data. Another main difference is that the FZARTMAP uses supervised learning. A standard ARTMAP uses unsupervised learning, hence it forms its own appropriate classes from the input data. In doing this it tries to form templates which are compared to the input. During learning, if the similarity between the input and a template is strong, the ART will adjust this template. If the similarity to all of the templates is weak, it will create a new template. The ”minimum degree of similarity” when deciding whether to create a new class is user definable and controlled by the *vigilance parameter*. The FZARTMAP has the capability of increasing the vigilance above this value to obtain the best possible classification. The basic idea behind the FZARTMAP is to use two ART networks (a and b) and simultaneously feed the input respectively the desired output through these. If the networks are not connected they would each create their own classes but when connected via the MTS the FZARTMAP will create as many internal classes as needed to create an almost total fit of the training data. The FZARTMAP then assigns these internal classes to a correct output. In our experiments, the output data form a one-of-N-coded category. In this special case, the FZARTMAP can be reduced to consist only of the ARTa and the MTS. For a detailed description of the fuzzy ARTMAP see the articles by Carpenter [4,5,6].

5.1 Experiments

In the experiments we gave the network a maximum of 135 internal classes. The base vigilance was set to 0; the recode rate to 1, the choice parameter, to 0.001; the epoch size 1640; and number of iterations, to 16 300. All parameters were kept constant during learning, so the learning schedule for the FZARTMAP is not necessary. Other parameters such as the number of units and the number of learning iterations were varied. This resulted in slightly decreased performance, although still within the confidence interval of the best net.

5.2 Results

The results in Table 13 show that the forecast of the FZARTMAP is excellent in the training set. However, its generalization capability on the test set is the worst of the network paradigms tested. It, nevertheless, shows the same behaviour as the other nets, its predictions are very good except for classes 6 and 7 were all nets perform at their worst (see Table 22 in tha appendix).

Table 13: Performance of the FZARTMAP. The lines in boldface hold the 95% confidence intervals for the classification rates (lower bound, average, upper bound, respectively).

		Train set			Test set	
	Classif. rate	Average	Std. Dev.	Classif. rate	Average	Std. Dev.
	0.991	**0.994**	**0.002**	0.871	**0.875**	**0.005**
	0.995			0.879		
	0.994			0.868		
	0.994			0.878		
	0.995			0.877		
C.I. 95%	**0.990**	**0.994**	**0.997**	**0.865**	**0.875**	**0.884**

5.3 Discussion

The fuzzy ARTMAP is able to reproduce the learn set to any degree of accuracy. The generalization capability of the network for the given application, however, is not as good as that of the other network types that are discussed in this paper.

6. Back-propagation

The back-propagation network is one of the most widely used types of neural networks. A back-propagation network for classification consists of an input layer, one or more hidden layers, and an output layer. The number of units in the input and output layers equals the number of explanatory variables and the number of a priori classes, respectively. The number of hidden layers and units in the network is not directly defined by the structure of the problem. The researcher chooses values for these numbers depending on the complexity of the problem,. The number of hidden layers is usually set to one. Since there is no definite rule for estimating the number of units in the hidden layer, it is usually determined by trial and error.

The back-propagation network feeds the inputs to each unit in the first hidden layer; in these units the inputs are scaled by an individual weight and totaled. This weighted sum is applied to a transfer function, the result of which is a scaled unit output. The outputs in one layer are inputs for units in the next layer and this process is repeated foro each layer. The output from the net is compared with the desired output using an error function. The weights are then adjusted by a gradient descent algorithm that involves back-propagating the error to previous layers. This network type is described in more detail in [7]. The network developer has several choices such as the number of hidden layers and units, the size of the step during error correction for each layer (learning coefficient), the number of inputs to be presented between every weight update (epoch size), the transfer function, and the learning rule.

6.1 Experiments

In Fischer et. al. [8] a back-propagation network model that uses a pruning algorithm, epoch size 3, learning coefficient 0.8, and no momentum is described. Learning starts with 22 units in the hidden layer, eight of which are pruned away. A classification rate of 90% was achieved in their study. We reproduced this experiment with 14 hidden units, without a pruning. We used the learn schedule displayed in Table 14 to train a network with 14 units in the hidden layer. Epoch size 1 was selected for this experiment.

Table 14: Learn schedule for back-propagation neural network

Learn count	10000	30000	50000
Learning rate	0.30000	0.15000	0.03750
Momentum	0.40000	0.20000	0.05000

6.2 Results

Our simulations showed that an initial value of 0.3 for the learning coefficient gave the best performance. Table 15 present the average and the 95% confidence interval from five simulations.

Table 15: Performance of back-propagation network with 14 hidden units.

		Train set			**Test set**	
Run	**Classif. rate**	**Average**	**Std. Dev.**	**Classif. rate**	**Average**	**Std. Dev.**
1	0.920	**0.920**	**0.001**	0.893	**0.893**	**0.004**
2	0.918			0.888		
3	0.920			0.890		
4	0.921			0.896		
5	0.920			0.898		
C.I. 95%	**0.918**	**0.920**	**0.997**	**0.885**	**0.893**	**0.901**

Appendix Table 23 provides details of the number of correctly predicted observations in each class for the best individual net. As mentioned earlier we tried many unit sizes and also a different learning rule but they are not presented because these results were compatible and within the confidence interval of the winner.

6.3 Discussion

Our simulations show that although the back-propagation network is not the best net, its average performance is well inside the confidence interval of the best-performing LVQ net.

7. Radial Basis Function Network

The radial basis function network (RBFN) in this study is a combination of two network paradigms: unsupervised learing and supervised learning. The unsupervised phase used the inputs Euclidean distance to find centers of clusters with the mean of a K-mean clustering method. The nearest neighbour method (see chapter 8.1) was then used to estimate the width of a Gaussian transfer function. This method was followed by a softmax linear transfer function in the output layer. The RBFN gets it name from the radially symmetric patterns it creates from the inputs. For this pattern to result there must be a center which is a vector in the input space, a distance measure from the center to the input vectors, and a single variable transfer function which maps the output from the transfer function and thereby determines the output of the single unit. The weights estimated in this self-organizing phase are referred to as prototype weights. Finally when the unsupervised phase is over a standard delta rule can be used to perform supervised learning. A detailed description is given in [9].

7.1 Experiments

During our experiments we found that the following parameters gave the best results; 64 prototype weights, Euclidean distance measure, normalized cumulative delta rule, 90,000 learning iterations, a Gaussian transfer function in the hidden layer and softmax output. During experiments the optimal number of prototype weights were found to be the above mentioned. The values we used for training the best net are shown in Table 16. In this table the columns with momentum 0 correspond to the self-organizing phase in which the input to hidden layer weights are learned. The other columns correspond to the phase in which the hidden-to-output-layer weights are learned by a gradient descent algorithm. The first phase consisted of 49,200 iterations.

Table 16: Learn schedule for RBFN. Total number of learning iterations.

Learn count	12 300	24 600	36 900	49 200	59 200	79 200	90 000
Learning rate	0.3	0.15	0.075	0.0375	0.8	0.4	0.1
Cluster threshold	0.1	0.05	0.025	0.0	0.0	0.0	0.0
Momentum	0.0	0.0	0.0	0.0	0.4	0.2	0.05
Error tolerance	0.0	0.0	0.0	0.0	0.1	0.1	0.1

7.2 Results

During experiments the optimal number of prototype weights was found to be 64, but we also tried different epoch sizes with the delta-rule. Performance of the resulting nets did not differ that much from one another but the best performance came with parameters presented in Table 16. Table 17 shows that the RBFN is one of the best performing nets. It fails nevertheless as all other methods in correctly separating categories 6 and 7 as seen in Table 24 in the appendix. The confidence interval is in this case overlapped the confidence intervals of the other best nets (LVQ, SOM).

Table 17: Performance of radial basis function network with 64 prototype weights.

		Train set			Test set	
	Classif. rate	Average	Std. Dev.	Classif. rate	Average	Std. Dev.
	0.931	**0.930**	**0.003**	0.911	**0.907**	**0.005**
	0.924			0.899		
	0.931			0.909		
	0.932			0.905		
	0.929			0.910		
C.I. 95%	**0.923**	**0.930**	**0.936**	**0.897**	**0.907**	**0.916**

7.3 Discussion

The RBFN has the second best performance; this is not surprising given that it is similar to the LVQ net in that it resembles a SOM net with supervised learning. One reason for the improved performance may be that the learning rules in the RBFN are better suited for the relationships mirrored in our data set than in other supervised networks.

8. Statistical classifiers

In this section we present the results of four statistical methods used in our classification problem. The purpose of showing the results of these methods is to to compare the performance of the proposed neural classifiers with other widely accepted methods.

The classifiers covered in this section can be divided into parametric and nonparametric methods. All of the statistical methods discussed in this paper calculate a probability density estimation for each of the eight categories. The a posteriori probabilities are then defined by Bayes' theorem: they are defined by the a priori probability multiplied by the density estimate, and divided by the total density estimate for all categories.

8.1 Nonparametric classification methods

The nonparametric methods that have been examined are kernel methods and nearest neighbor methods. The kernel methods calculate the probability density estimations based on generalized distances from the out-of-sample observation of each training example in a category. In our experiments, normal kernels were used with radii ranging from 0.375 to 3. The a posterior probabilities are then calculated by means of Bayes' theorem, taking into account the a priori probabilities.

When classifying a new observation, the *k* nearest neighbor method uses only the *k* observations in the train set that are nearest to a generalized distance function. The probability density function for category *C* is then determined by the number of category *C* observations among the *k* nearest observations relative to the total number of category *C* observations in the train set and the a priori probability of an observation to have category *C*. The a posterior probability of category *C* is then defined by Bayes' theorem. In our experiments, *k* varied from 1 to 35. The a priori probabilities were chosen so that they would be proportional to the numbers of learning examples in each category.

8.1.1 Results

These methods can easily compete with the neural network classifiers. Only the best network types give a somewhat higher accuracy. Because the performance for even numbers of nearest neighbors was consistently lower than those for odd numbers, Figure 4 and Figure 5 shows the results.

Figure 4: Results of the kernel method (normal kernel)

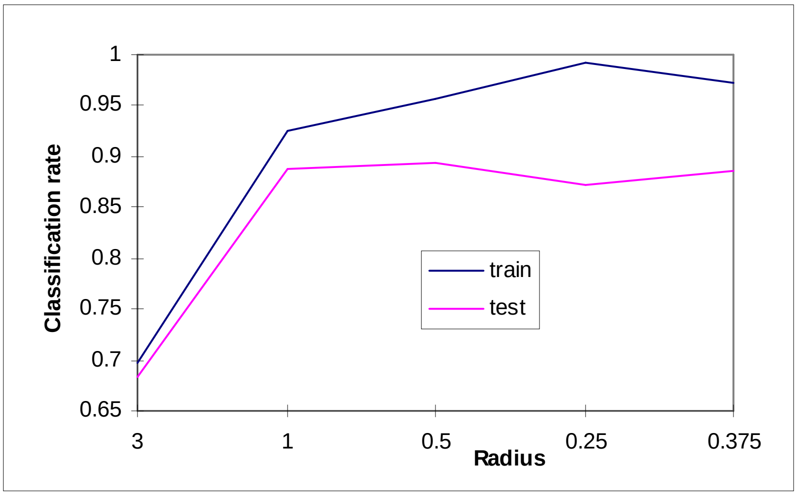
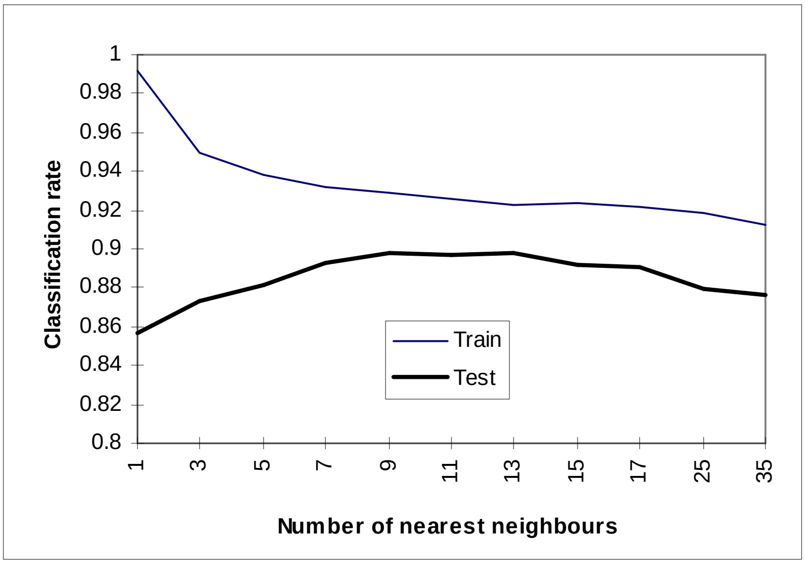


Figure 5: Results with the k-nearest neighbours method.



8.2 Parametric methods

The third and fourth statistical methods are parametric discriminant analysis methods, also known as the Gaussian maximum likelihood methods. They are well-established classification tools that have been implemented in commercial statistical packages. The software used in our study is Minitab. Both linear and quadratic discriminant analyses have been employed. The results obtained with these methods are given in Table 18. The classification matrices for the statistical method that performed best, linear discriminant analysis, are displayed in Table 26 in the appendix.

8.3 Results and discussion

Table 18: Classification rates for the different statistical methods used.

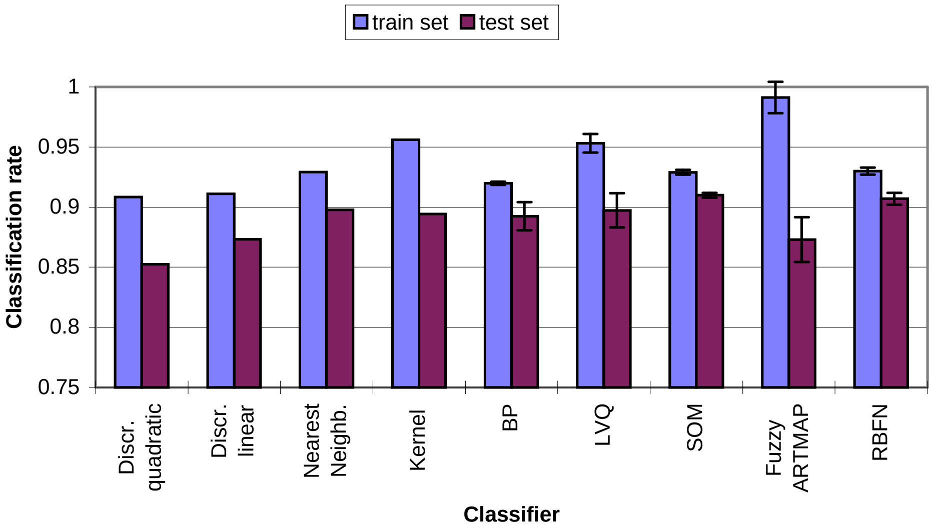
Method	train set	test set
Nearest Neighbor	0.929	0.898
Kernel classifier	0.956	0.894
Linear discriminant analysis	0.911	0.873
Quadratic discriminant analysis	0.909	0.852

Results from the distribution-free show that these methods can compete with the neural network classifiers. The parametric methods give significantly lower results than the non-parametric statistical methods. This is in accordance with the remark in Section 1 that neural network methods are best compared with nonparametric methods.

9. Comparison of the tested classifiers

In summary, the best network type is the self organizing feature map, followed by the radial basis function network (see Figure 6).

Figure 6: Overview of all classifiers in this paper, with error bars for the neural classifiers.



9.1 Discussion

Almost all of the methods achieved a classification rate that was close to 90%. Parametric statistical methods and the fuzzy ARTMAP neural network differed from the best performing methods in this aspect. All of the statistical methods assumed that the inputs of each category had a multivariately normal distribution; something that does not apply to the non-parametric methods. One reason for the bad performance of the fuzzy ARTMAP could be its internal structure. It is very good at learning a train set and should be good at predicting as well. Why this is not the case may be that when new data differ greatly from the data in the train set, it cannot correctly assign them to the internal classes and fails.

10. Input Dependency

The complexity of the problem is determined by, among other things, the dimension of the input space.

In the data set at hand, the three input vector components that correspond to the visual part of the spectrum show high correlation. This can be seen in Table 19. To determine if all of the inputs are necessary, we left out one spectral band and trained an LVQ net with the remaining five-dimensional inputs. We also trained a net with four inputs: the infrared channels and the average of the three visible channels.

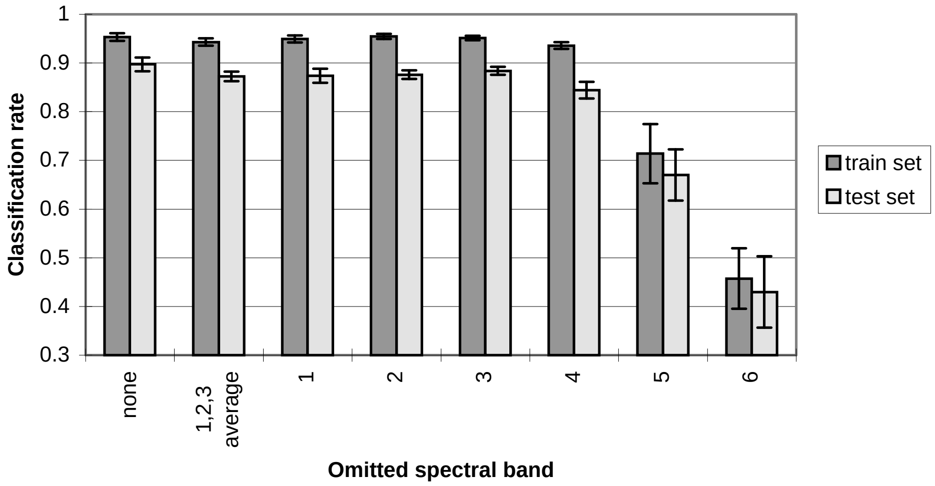
Table 19: Correlation matrix for the six spectral bands.

	C1	C2	C3	C4	C5
C2	0.959				
C3	0.940	0.961			
C4	-0.487	-0.367	-0.376		
C5	0.196	0.312	0.321	0.644	
C6	0.685	0.734	0.770	0.101	0.795

The network used for these experiments was a 200-unit learning vector quantization network (see Section 4) trained for 41 000 iterations. The learn schedule was identical to that listed in Table 5.

Figure 7 displays the results with error bars. Other tested network types show similar results.

Figure 7: Effect of reducing the dimension of the input space.



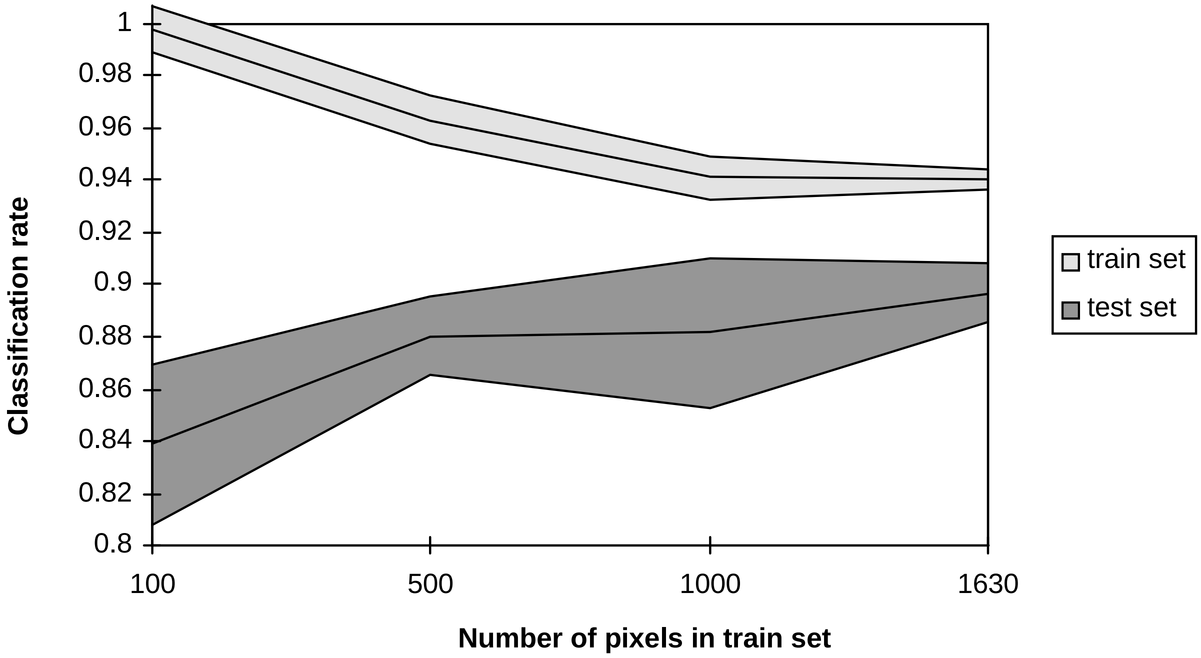
These results indicate a very small decrease in accuracy when one of the visisble channels is left out or even when the three visible channels are replaced by their average, leaving only four inputs and thereby considerably reducing the network complexity. The infrared channels are more important, which is in accordance with the values shown in the covariance matrix.

11. Size of the Training Set

Determining the class of a pixel by sight is a cumbersome process. Therefore, it is important to know how many examples are needed to have a network that is capable of generalization. We tested four different train set sizes. The network used for these experiments was a 200-unit learning vector quantization network (see Section 4) trained for 41 000 iterations. The learn schedule was identical to that listed in Table 5.

Figure 8 shows that 500 elements in the training set give results that are almost as good as those obtained by training with the whole set. However, the results tend to improve as the size of the training set increases. Experiments with larger data sets are needed to arrive at definite conclusions.

Figure 8: Effect of the size of the training set on the performance of an LVQ net.



12. Conclusion

All methods compared in this paper produce similar results except for two methods: discriminant analysis and the fuzzy ARTMAP artificial neural network which achieved lower classification rates. The best-performing network architecture, which was the self-organizing feature map correctly classified 91 percent which is slightly better than the radial basis function network's 90%. The overall performance is very good and the results are encouraging.

However, it should be stressed that these data were gathered with the so-called one-site condition, which means that all examples for an a priori category are taken from a single site. This results in overly homogeneous data, which are not available in many practical situations. Furthermore, distribution-free statistical methods achieved over 89% classification accuracy which is very close to the neural classifiers.

The results in this paper are useful as a starting point for studies with more heterogeneous data. Future research is needed to determine if the results in this study achieved here are general and can be applied to other data. That is, will these methods perform in a similar way when used on other data sets?

References

[1] Software Manual, multiple authors (1995). Neural computing: a technology handbook for Professional II/PLUS and NeuralWorks Explorer. NeuralWare, Inc., Pittsburgh.

[2] Kohonen, T. (1988). Self-organization and associative memory. Second edition. Springer-Verlag, New York.

[3] Kohonen, T. et al. (1988). Statistical pattern recognition with neural networks: benchmark studies. Proceedings of the Second Annual IEEE International Conference on Neural Networks. Volume 1.

[4] Carpenter, G. A., Grossberg, S., and Reynolds, J. H. (1991). ARTMAP: Supervised real-time learning and classification of nonstationary data by a self-organizing neural network. Neural Networks, 4, pp. 565-588.

[5] Carpenter, G. A., Grossberg, S., and Rosen, D. B. (1991). Fuzzy ART: Fast stable learning and categorization of analog patterns by an adaptive resonance system. Neural Networks 4, pp. 759-771.

[6] Carpenter, G. A., Grossberg, S., Markuzon, N., Reynolds, J. H., and Rosen, D. B. (1992). Fuzzy ARTMAP: An adaptive resonance architecture for incremental learning of analog maps. IJCNN June 1992, Baltimore, III, pp. 309-314.

[7] Rumelhart, D. E., McClelland, J. L. (editors) (1986). Parallel distributed processing: explorations in the microstructure of cognition. Volume I, Chapter 8. MIT Press, Cambridge

[8] Fischer, M.M, Wessels, J. et. al.(1996) Intermediate report on the research project. The potential of Neurocomputing in Regional Science: exploratory analysis of new methodoligies in selected application fields. IIASA

[9] Moody, J., Darken, C. J., (1989) Fast Learning in Networks of Locally Tuned Processing Units, Neural Computing 1,. pp. 281-294.

Appendix

Table 20: Classification matrices of the best SOM net at a constant learning rate.

			Classifier's categories (train set)						
Ground truth categories	C1	C2	C3	C4	C5	C6	C7	C8	Total
C1	157	9	0	1	0	0	0	0	167
C2	1	282	0	2	0	0	0	0	285
C3	0	0	126	0	0	2	0	0	128
C4	2	1	0	393	6	0	0	0	402
C5	0	0	2	5	95	0	0	0	102
C6	0	0	1	0	0	277	12	6	296
C7	0	0	0	0	0	63	90	0	153
C8	0	0	0	0	0	5	0	102	107
Total	160	292	129	401	101	347	102	108	1640

	Classifier's categories (test set)								
Ground truth categories	C1	C2	C3	C4	C5	C6	C7	C8	Total
C1	78	5	0	0	0	0	0	0	83
C2	1	141	0	0	0	0	0	0	142
C3	0	0	63	0	0	1	0	0	64
C4	3	2	0	194	1	0	0	0	200
C5	0	3	0	0	49	0	0	0	52
C6	0	0	0	0	1	127	20	0	148
C7	0	0	0	0	0	29	48	0	77
C8	0	0	0	0	0	5	0	49	54
Total	82	151	63	194	51	162	68	49	820

Table 21: Classification matrices of the best performing LVQ network.

Ground truth categories	Classifier's categories (train set)								
	C1	C2	C3	C4	C5	C6	C7	C8	Total
C1	163	4	0	0	0	0	0	0	167
C2	2	283	0	0	0	0	0	0	285
C3	0	0	127	0	1	0	0	0	128
C4	1	2	0	390	9	0	0	0	402
C5	0	0	0	0	102	0	0	0	102
C6	0	0	1	0	1	266	7	21	296
C7	0	0	0	0	1	69	83	0	153
C8	0	0	0	0	0	0	0	107	107
Total	166	289	128	390	114	335	90	128	1640

Ground truth categories	Classifier's categories (test set)								
	C1	C2	C3	C4	C5	C6	C7	C8	Total
C1	81	2	0	0	0	0	0	0	83
C2	1	139	0	0	2	0	0	0	142
C3	0	0	64	0	0	0	0	0	64
C4	2	3	0	189	3	0	0	3	200
C5	0	1	0	0	51	0	0	0	52
C6	0	0	0	0	2	119	17	10	148
C7	0	0	0	0	0	33	44	0	77
C8	0	0	0	0	0	1	0	53	54
Total	84	145	64	189	58	153	61	66	820

Table 22: Classification matrices of the best fuzzy ARTMAP.

			Classifier's categories (train set)						
Ground truth categories	C1	C2	C3	C4	C5	C6	C7	C8	Total
C1	167	0	0	0	0	0	0	0	167
C2	0	285	0	0	0	0	0	0	285
C3	0	0	128	0	0	0	0	0	128
C4	0	0	0	402	0	0	0	0	402
C5	0	0	0	0	102	0	0	0	102
C6	0	0	0	0	0	291	3	2	296
C7	0	0	0	0	0	5	148	0	153
C8	0	0	0	0	0	0	0	107	107
Total	167	285	128	402	102	296	151	109	1640

			Classifier's categories (test set)						
Ground truth categories	C1	C2	C3	C4	C5	C6	C7	C8	Total
C1	79	4	0	0	0	0	0	0	83
C2	3	133	5	0	1	0	0	0	142
C3	0	0	63	0	0	1	0	0	64
C4	2	2	0	194	1	0	0	1	200
C5	0	4	0	0	48	0	0	0	52
C6	0	0	0	0	1	99	44	4	148
C7	0	0	0	0	0	24	53	0	77
C8	0	0	0	0	0	3	0	51	54
Total	84	143	68	194	51	127	97	56	820

Table 23: Classification matrices of the best back-propagation net.

			Classifier's categories (train set)						
Ground truth categories	C1	C2	C3	C4	C5	C6	C7	C8	
C1	158	7	0	2	0	0	0	0	167
C2	2	282	0	1	0	0	0	0	285
C3	0	0	127	0	0	1	0	0	128
C4	3	0	0	392	7	0	0	0	402
C5	0	0	2	5	95	0	0	0	102
C6	0	0	1	0	0	259	24	12	296
C7	0	0	0	0	0	60	93	0	153
C8	0	0	0	0	0	4	0	103	107
Total	163	289	130	400	102	324	117	115	1640

			Classifier's categories (test set)						
Ground truth categories	C1	C2	C3	C4	C5	C6	C7	C8	
C1	77	4	0	2	0	0	0	0	83
C2	1	139	0	0	2	0	0	0	142
C3	0	0	64	0	0	0	0	0	64
C4	1	2	0	192	2	0	0	3	200
C5	0	2	0	0	50	0	0	0	52
C6	0	0	0	0	1	113	31	3	148
C7	0	0	0	0	0	29	48	0	77
C8	0	0	0	0	0	1	0	53	54
Total	79	147	64	194	55	143	79	59	820

Table 24: Classification matrices of the best radial basis function net.

			Classifier's categories (train set)						
Ground truth categories	C1	C2	C3	C4	C5	C6	C7	C8	
C1	164	3	0	0	0	0	0	0	167
C2	2	282	0	1	0	0	0	0	285
C3	0	0	127	0	0	0	1	0	128
C4	0	0	0	395	7	0	0	0	402
C5	0	0	0	5	97	0	0	0	102
C6	0	0	1	0	0	269	17	9	296
C7	0	0	0	0	0	59	94	0	153
C8	0	0	0	0	0	8	0	99	107
Total	166	285	128	401	104	336	112	108	1640

			Classifier's categories (test set)						
Ground truth categories	C1	C2	C3	C4	C5	C6	C7	C8	
C1	81	2	0	0	0	0	0	0	83
C2	0	137	5	0	0	0	0	0	142
C3	0	0	64	0	0	0	0	0	64
C4	1	3	0	196	0	0	0	0	200
C5	0	3	0	0	49	0	0	0	52
C6	0	0	0	0	2	121	25	0	148
C7	0	0	0	0	0	29	48	0	77
C8	0	0	0	0	0	3	0	51	54
Total	82	145	69	196	51	153	73	51	820

Table 25: Classification matrices of a classifier based on linear discriminant analysis.

Ground truth categories			Classifier's categories (train set)						
	C1	C2	C3	C4	C5	C6	C7	C8	Total
C1	163	3	0	1	0	0	0	0	167
C2	2	280	0	0	3	0	0	0	285
C3	0	0	123	0	2	3	0	0	128
C4	7	2	0	381	12	0	0	0	402
C5	0	1	1	0	100	0	0	0	102
C6	0	0	0	0	1	247	27	21	296
C7	0	0	0	0	0	60	93	0	153
C8	0	0	0	0	0	0	0	107	107
Total	172	286	124	382	118	310	120	128	1640

Ground truth categories	Classifier's categories (test set)								
	C1	C2	C3	C4	C5	C6	C7	C8	Total
C1	81	2	0	0	0	0	0	0	83
C2	2	138	0	0	2	0	0	0	142
C3	0	1	62	0	0	1	0	0	64
C4	4	3	0	183	9	0	0	1	200
C5	0	2	0	0	50	0	0	0	52
C6	0	0	0	0	2	100	36	10	148
C7	0	0	0	0	0	29	48	0	77
C8	0	0	0	0	0	0	0	54	54
Total	87	146	62	183	63	130	84	65	820

Table 26: Classification matrices of the K-nearest neighbor method.

Ground truth categories			Classifier's categories (train set)						
	C1	C2	C3	C4	C5	C6	C7	C8	Total
C1	162	4	0	1	0	0	0	0	167
C2	0	283	0	0	1	0	0	0	285
C3	0	0	127	0	0	0	0	0	128
C4	3	3	0	388	7	0	0	0	402
C5	0	1	1	0	98	0	0	0	102
C6	0	0	0	0	0	259	16	20	296
C7	0	0	0	0	0	52	101	0	153
C8	0	0	0	0	0	1	0	106	107
Total	165	291	128	389	106	312	117	126	1640

Ground truth categories	Classifier's categories (test set)								
	C1	C2	C3	C4	C5	C6	C7	C8	Total
C1	81	2	0	0	0	0	0	0	83
C2	0	141	0	0	1	0	0	0	142
C3	0	1	62	0	0	1	0	0	64
C4	1	2	0	192	2	0	0	0	200
C5	0	2	0	0	50	0	0	0	52
C6	0	0	0	0	1	109	29	9	148
C7	0	0	0	0	0	29	48	0	77
C8	0	0	0	0	0	1	0	53	54
Total	82	148	62	192	54	140	77	62	820

Figure 9:

The SOM-Classified Image

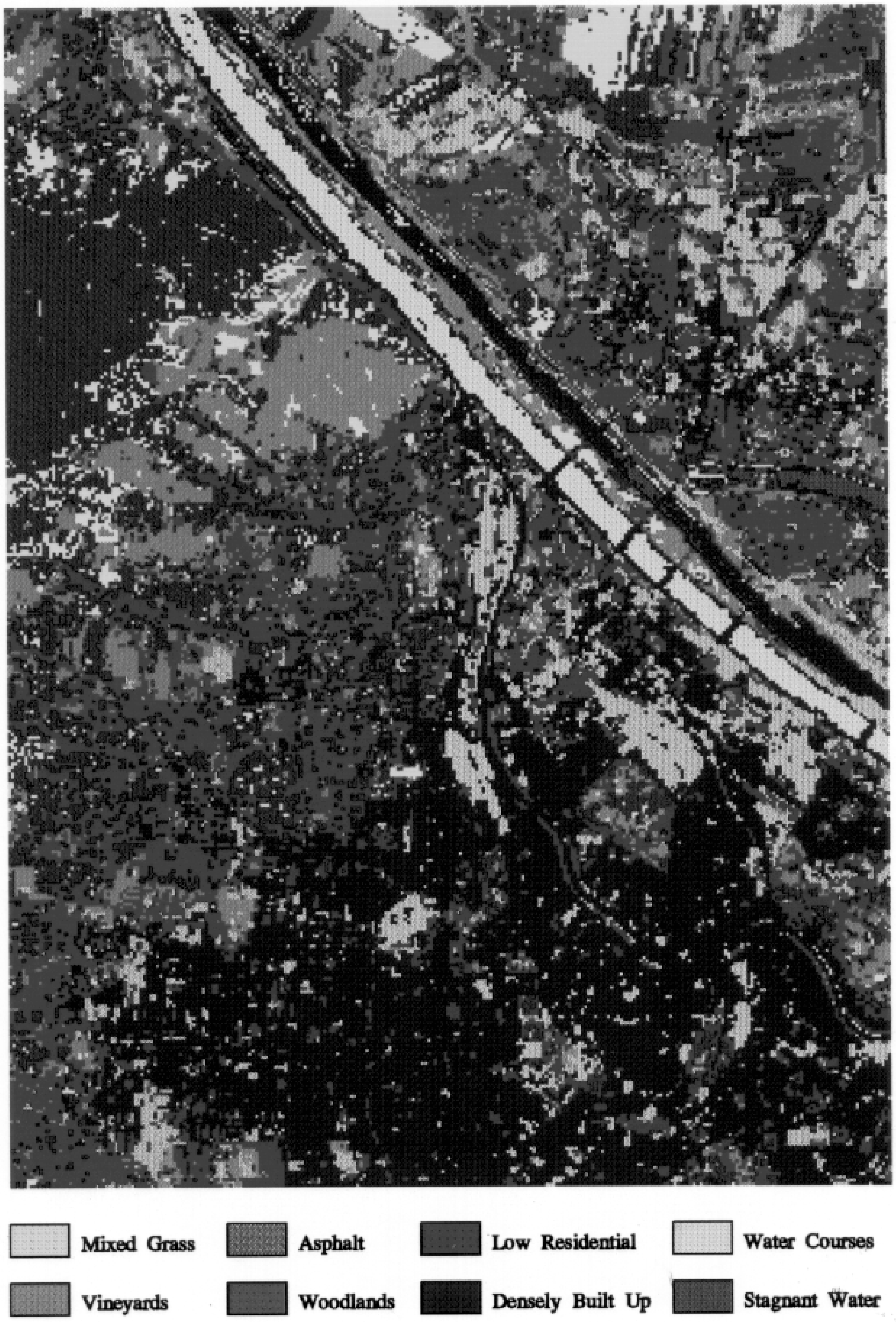


Figure 10:

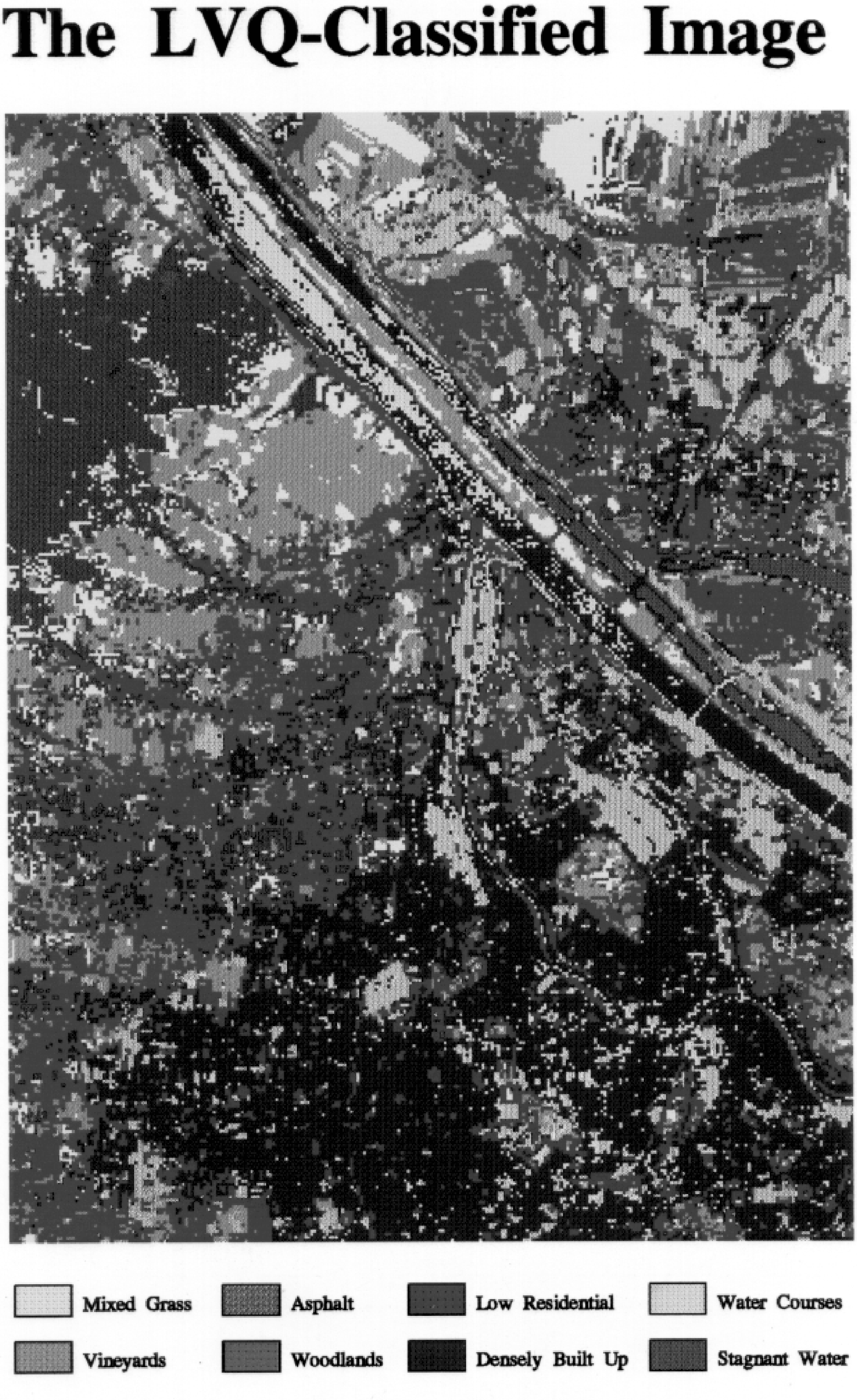
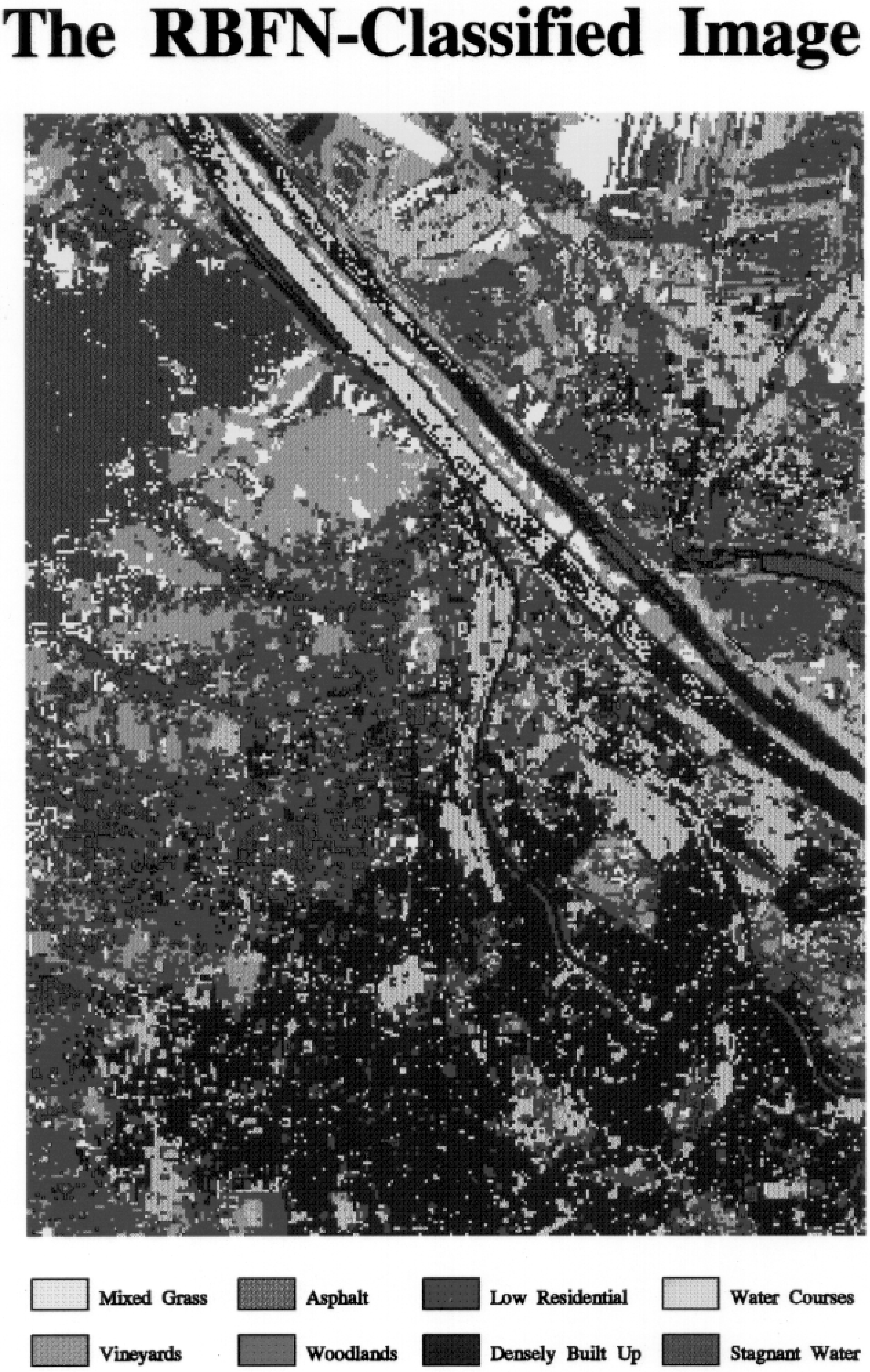

The LVQ-Classified Image
Mixed Grass
Asphalt
Low Residential
Water Courses
Vineyards
Woodlands
Densely Built Up
Stagnant Water

Figure 11:


The RBFN-Classified Image
Mixed Grass
Asphalt
Low Residential
Water Courses
Vineyards
Woodlands
Densely Built Up
Stagnant Water