

8-1-1991

CONSENSUAL NEURAL NETWORKS

Jon A. Benediktsson

University of Iceland, Department of Electrical Engineering & Laboratory for Information Technology and Signal Processing

Okan K. Ersoy

Purdue University School of Electrical Engineering

Philip H. Swain

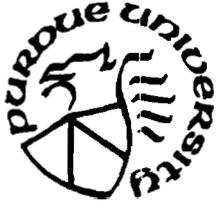
Purdue University School of Electrical Engineering

Follow this and additional works at: <http://docs.lib.purdue.edu/ecetr>

Benediktsson, Jon A.; Ersoy, Okan K.; and Swain, Philip H., "CONSENSUAL NEURAL NETWORKS" (1991). *ECE Technical Reports*. Paper 319.

<http://docs.lib.purdue.edu/ecetr/319>

This document has been made available through Purdue e-Pubs, a service of the Purdue University Libraries. Please contact epubs@purdue.edu for additional information.



Consensual Neural Networks

Jon A. Benediktsson
Okan K. Ersoy
Philip H. Swain

TR-EE 91-37
August 1991

CONSENSUAL NEURAL NETWORKS

Jon A. Benediktsson*, Okan K. Ersoy** and Philip H. Swain**

TR-EE 91-37

August 1991

* Department of Electrical Engineering

and

Laboratory for Information Technology and Signal Processing

University of Iceland

Hjardarhaga 2-6

107 Reykjavik, Iceland.

** School of Electrical Engineering

and

Laboratory for Applications of Remote Sensing

Purdue University

W. Lafayette, IN 47907, U.S.A.

ACKNOWLEDGEMENTS

The Colorado data set was originally acquired, **preprocessed** and loaned to us by Dr. Roger Hoffer of Colorado State University. **Access** to the data set is gratefully acknowledged.

The research was supported in part by the National Aeronautics and Space Administration through Grant No. NAGW-925.

This work was supported in part by NASA Grant No. NAGW-925 "Earth Observation Research - Using Multistage **EOS-like** Data" [Principal Investigators: David A. Landgrebe and Chris **Johannsen**].

ABSTRACT

A new neural network architecture is proposed **and** applied in classification of data from multiple sources. The new **architecture** is called a consensual neural network **and** its relation to **hierarchical** and ensemble neural networks is discussed. The **consensual neural network** architecture is based on statistical consensus **theory** and **involves** using non-linearly transformed input data. The input data are transformed several **times** and the different transformed data are applied as if they were independent inputs. The independent inputs are **classified** using stage neural networks and the **outputs** from the stage: networks are then weighted and combined to make a decision. Experimental results based on remote sensing data and geographic data are given. The **performance** of the consensual **neural network architecture** is compared to that of a two-layer conjugate-gradient backpropagation neural network. The results with the proposed neural network architecture compare favourably to the **backpropagation** method in terms of classification accuracy.

1. INTRODUCTION

The recent resurgence of research in neural **networks** has resulted in the development of new and improved **neural network** models. These new models have been trained successfully to classify complex data. In the remote sensing community, the **question** of how well **neural network** models perform as classifiers is very important. In **previous** papers [1],[2], it has been shown that neural networks

compared well to statistical classification methods in **classification** of multisource remote **sensing/geographic** data and very-high-dimensional data. The neural network models were superior to the statistical methods in terms of overall classification accuracy of **training** data. However, statistical methods based on *consensus* from several data sources outperformed the neural networks in **terms of overall** classification accuracy of test data. Thus it would be very desirable to combine certain aspects of the statistical consensus theory approaches and the neural network models. **However**, it is very difficult to implement statistics in neural networks [3].

In this report, *consensual neural networks* are proposed and implemented as *stage-wise* neural network **algorithms**. These network models do not use prior statistical **information** but are somewhat analogous to the statistical consensus theory approaches. A short overview of consensus theory is given in the **next** section followed by a discussion of neural networks as **related** to the proposed consensual neural networks. The consensual neural networks are then addressed in some detail and experimental results using these networks are presented.

2. CONSENSUS THEORY

Consensus theory [3],[4],[5],[6] is a well-established research field involving procedures for combining estimated probability distributions of multiple data sources under the assumption that the data sources are Bayesian. In most consensus theoretic methods, the

data from each source are at first classified into a **source-specific number of *data classes*** [1]. The information from the sources is then aggregated by a global membership function and the data are classified according to the usual maximum selection rule into a **user-specified number of *information classes***. The **combination** formula obtained in consensus theory is called a ***consensus rule***. Several consensus rules have been proposed. Probably the **most commonly used** consensus rule is the ***linear opinion pool*** which has the following form for the information class ω_j if n data sources are used:

$$C(X) = \sum_{i=1}^n \alpha_i p(\omega_j | x_i) \quad (1)$$

where $X = [x_1, \dots, x_n]$ is the vector of multichannel **data** values at a **pixel**, $p(\omega_j | x_i)$ is a source-specific posterior probability and α_i s ($i = 1, \dots, n$) are source-specific weights which control the relative influence of the data sources. The weights are associated with the sources in the global membership function to express quantitatively our **confidence** in each source [3]. The linear opinion pool is simple but **has** several shortcomings, **e.g.**, it is not externally Bayesian since it is not derived from class-conditional probabilities using Bayes' rule. Another consensus rule which overcomes the shortcomings associated with the linear opinion pool is the ***logarithmic opinion pool***:

$$L(X) = \prod_{i=1}^n (p(\omega_j | x_i))^{\alpha_i} \quad (2)$$

The logarithmic opinion pool has performed well in classification of **data** from multiple sources [3],[4].

It is desirable to implement consensus theoretic approaches in **neural** networks: consensus theory has the goal of **combining** several **opinions**, and a collection of different neural networks should be more accurate than a single network in classification. It is important to note that neural networks have been shown to approximate **class**-conditional probabilities, $p(\omega_j|x_j)$, at the output in the **mean** square **sense** [7]. Using this property of neural networks, it becomes possible to implement consensus theory in the networks.

3. NEURAL NETWORK METHODS

A neural network is an interconnection of neurons, where a neuron can be described in the following way: A neuron receives input signals X_j , $j = 1,2,\dots,N$, which represent the activity at the input or **the** momentary frequency of neural impulses delivered by another neuron to this input [8]. In the simplest formal model of a neuron, the **output** value of the neuron, o , is often approximated by

$$o = K \phi\left(\sum_{j=1}^N w_j x_j - \theta\right) \quad (3)$$

where K is a constant and ϕ is a non-linear function, e.g., the threshold function which takes the value 1 for positive arguments and 0 (or -1) for negative arguments. The W_j are called synaptic *efficacies* or weights, and θ is a threshold. A single layer neural network, only has one layer of weights; a multilayer network has a number of such layers [9]. In the neural network approach to pattern recognition, the neural network operates as a black box which receives a set of input vectors x (observed signals) and produces responses O_i from its output neurons i ($i = 1, \dots, L$ where L depends on the number of information classes). A common output representation used in neural network theory is that the outputs are either $O_i = 1$, if neuron i is active for the current input vector x , or $O_i = 0$ (or -1) if it is inactive. In supervised learning the weights are learned through an adaptive (iterative) training procedure in which a set of training samples is presented to the input (Figure 1). The network gives an output response for each sample. The actual output response is compared to the desired response for the input and the error between the desired output and the actual output is used to modify the weights in the neural network. The training procedure ends when the error is reduced to a prespecified threshold or it cannot be minimized any further. Then all of the data to be classified are fed into the network to perform the classification, and the network provides at the output the class representation for each pixel.

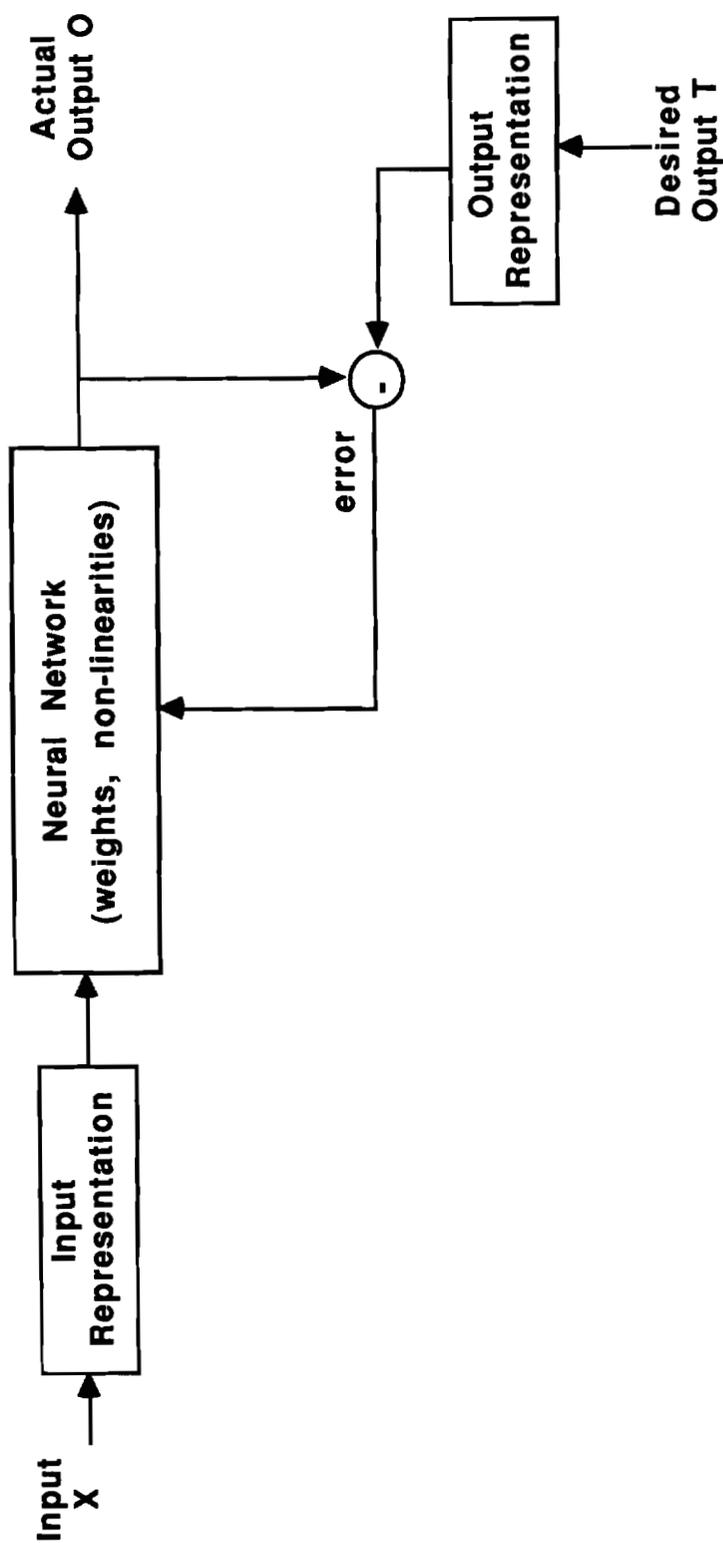


Figure 1. Schematic Diagram of Supervised Neural Network Training Procedure

3.1 Neural Networks with Parallel Stages

Implementing consensus theory in neural **networks** may be achieved by using a collection of neural networks. The parallel **self-organizing** hierarchical neural network (PSHNN) proposed by Ersoy and Hong [10] is a neural network which is in some respects related to **the** consensual neural network to be proposed here. The PSHNN involves a self-organizing number of stages, similar to a multilayer **neural** network. Each stage can be a particular neural network, here **referred** to as a stage neural network (SNN). Unlike a multilayer network, each SNN is essentially independent of the other SNNs in the **sense** that each SNN does not receive its input directly from the previous SNN. At the output of each SNN, there is an error detection scheme. If an input vector is rejected, it goes through a non-linear transformation before being input to the next SNN. This property is distinct from conventional neural networks.

Valafar and Ersoy [11] proposed a parallel, self-organizing, consensual neural network (PSCNN) which is related to **the** PSHNN [10]. The PSCNN uses non-linear transformations of the input data and creates **accept/reject** boundaries for each SNN in a similar **fashion** to the PSHNN. Pre- and post-voting are **used** to make decisions with the SNNs. The post-voting is somewhat similar to error boundaries in the PSHNN, but is not related to consensus **theory**.

Nilsson [12] proposed his *committee machines* as **an** attempt to **formulate** a multilayer neural network which could classify

complicated data. The committee machines are related to the **consensus** neural networks proposed here. However, the committee **machines** are not based on consensus theory and all the stages use the same inputs. The committee machines are an attempt to design a **multilayer** neural network by using one-layer networks.

Hansen and Salamon discussed the application of an ensemble of **multilayer** neural networks [13]. Their ensemble consists of several **SNNs** but each SNN receives the same input data similar to Nilsson's committee machines. Each SNN is based on the **backpropagation** network and the weights in different SNNs are **initialized** differently in order to avoid the same local **minima** for all of the networks. The ensemble network makes the final decision (classification) based on the majority vote from **all** the networks. The architecture in [13] is not based on consensus theory and does not use the capability of changing the input data through non-linear transformations.

The approach taken here is to use the data as separate and **distinct** inputs obtained through non-linear transformations of the **input** data, and to base the design of the total network on consensus theory.

3.2 The Consensual Neural Network

A block diagram for the proposed consensual **neural** network (CNN) architecture is shown in Figure 2. Each stage **neural** network (**SNN**) has the number of output neurons equal to the number of

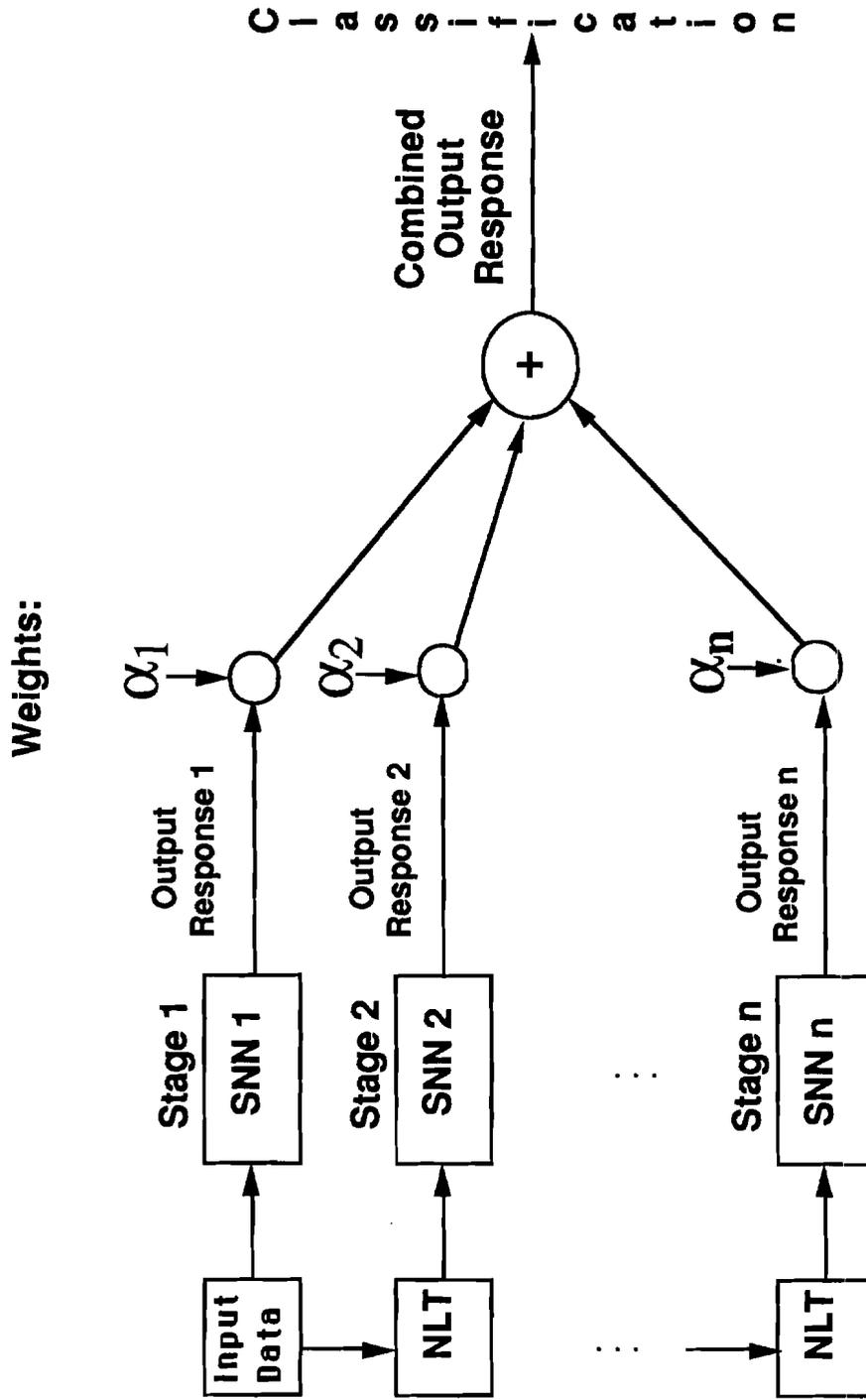


Figure 2. The Consensus Neural Network - Sum (CNNS) Architecture

information classes and is trained for a **fixed** number of iterations or **until** the training procedure converges. When the training of the first **stage** is complete, the classification error is computed. Then another stage is created. The input data to the second stage are obtained by a non-linear transform (NLT) of the original input vectors. This stage is trained in a fashion similar to the first stage. **When** the training of the second stage is complete, the *consensus* for the **SNNs** is computed. The consensus is obtained by **taking class-specific** weighted averages of the output responses of the SNNs using source-specific weights [3] similar to the ones in equations (1) and (2). Error detection is then performed and the *consensual classification error* is computed.

The CNN is self-organizing in the following sense: If the **consensual** classification error is lower than the classification error for the first stage, another stage is created and trained in a fashion similar to the previous stages, but with another non-linear **transformation** of the input data. Stages are added to the consensual **neural** network in this manner as long as the consensual classification error decreases or a tolerance limit is **reached**. If the consensual classification error does not decrease or is **lower** than the tolerance limit, the training is stopped. Using this architecture it can be guaranteed that the **CNNs** should do no worse than single stage networks, at least in training. To guarantee such performance in classification of test data, cross-validation methods can be used [14]. Also, it is easy to show [13] that if all the **networks** in a collection of neural networks arrive at the correct classification with a likelihood $1-p$ and the networks make independent errors, the chances of seeing exactly k errors among N copies of the **network** is:

$$\binom{N}{k} p^k (1-p)^{N-k} \quad (4)$$

which gives the following likelihood of a sum of **network** outputs being in error:

$$\sum_{k>N/2} \binom{N}{k} p^k (1-p)^{N-k} \quad (5)$$

which is monotonically decreasing in N if $p < 1/2$. Thus, using a collection of networks reduces the expected **classification** error if the :networks have equal weights and make independent **errors**.

Here we propose two versions of the CNN. The CNN in Figure 2 is called CNN - Sum (CNNS) and is a consensual **neural network version** of the linear opinion pool. The CNN - Product (CNNP) shown in Figure 3 is a consensual neural network version of the logarithmic opinion pool. Both CNNs combine the information **from** distinct inputs and can be considered neural network implementations of the **consensus** rules in equations (1) and (2). In contrast **to** the data sources usually referred to in multisource classification, the inputs here consist of non-linearly transformed data which have been transformed several times from the raw data. In neural networks it is very important to find the "best" representation of **input** data; the consensual method attempts to average over the results from several input representations. Also, in the consensual neural networks, **classification** of test data can be done in parallel, with all stages receiving data simultaneously, which makes this **method** attractive

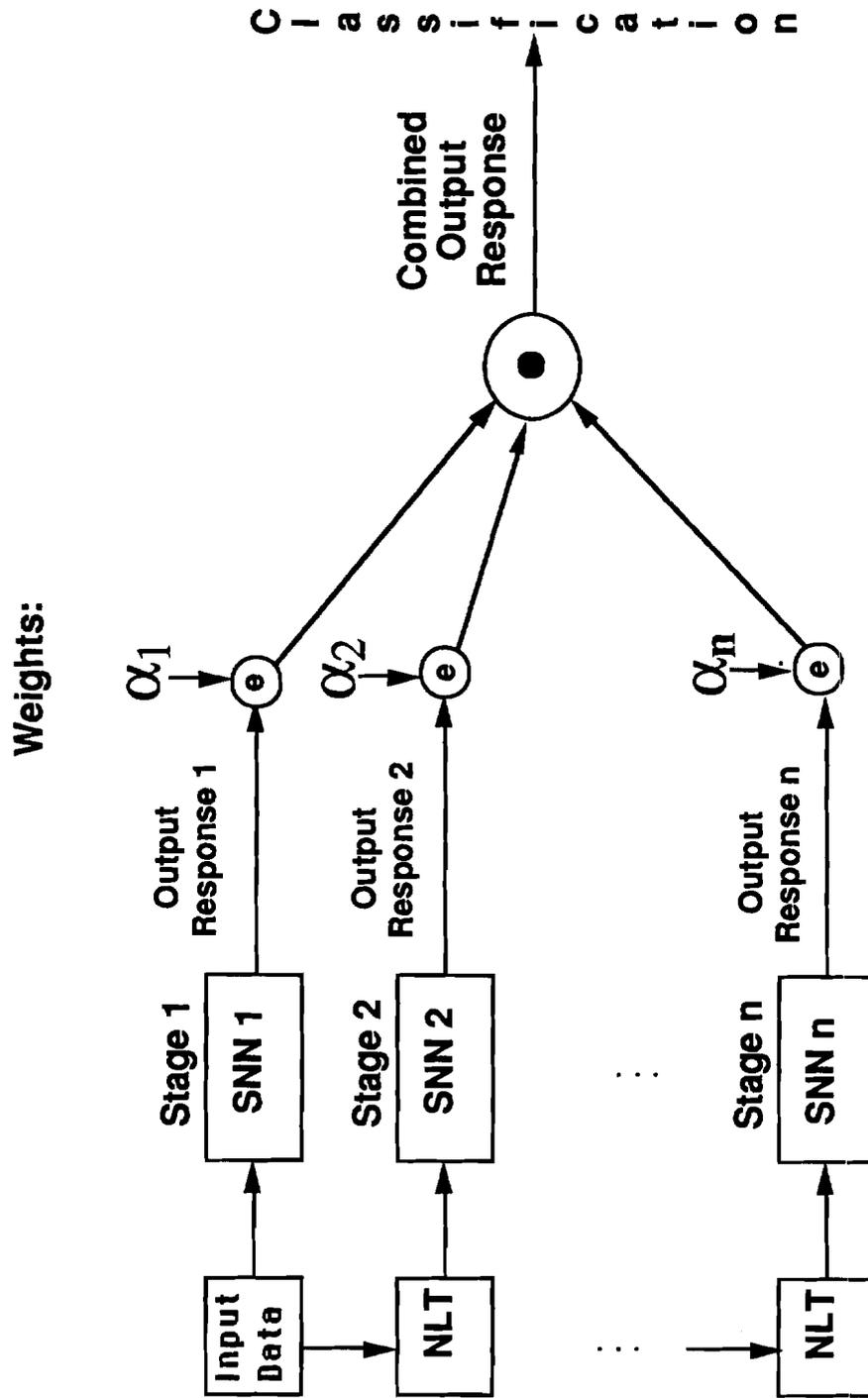


Figure 3. The Consensual Neural Network - Product (CNNP) Architecture

for **implementation** on parallel machines. Learning can **also** be made parallel, once the number of stages is determined.

The CNNs presented here are related to the PSHNN in the sense that both algorithms use stage networks. However, there are two major differences between the CNNs and the PSHNN. First, the **CNNs** non-linearly transform all the data whereas the PSHNN only propagates **misclassified** samples to the next stage and non-linearly transforms those samples. Secondly, the CNNs weight the outputs of the different SNNs whereas no weighting is done in **the** PSHNN. These properties of the CNNs are important since the **CNNs** need no rejection scheme at the outputs of the **SNNs**, but weight the outputs instead. The selection of non-linear transformations and weights for the **CNNs** are discussed below.

3.2.1 Non-Linear Transformations

The major source of classification error in single-stage neural networks is the linear **nonseparability** of the classes. To reduce or eliminate classification errors, it is desirable to find a **transformation which** maps the input vectors into another set of vectors that can be classified more accurately. A variety of schemes can be **used** in the **CNNs** to transform the data.

In the experiments performed here the input vectors were represented by the Gray code. The Gray code representation can be derived from the binary code representation in the following

manner: If $\mathbf{b}_1 \mathbf{b}_2 \dots \mathbf{b}_n$ is a code word in an n - bit binary code, the corresponding Gray code word $\mathbf{g}_1 \mathbf{g}_2 \dots \mathbf{g}_n$ is obtained by the rule:

$$\mathbf{g}_1 = \mathbf{b}_1$$

$$\mathbf{g}_k = \mathbf{b}_k \oplus \mathbf{b}_{k-1} \quad k \geq 2$$

where \oplus is modulo-two addition. One simple possibility for a **non-linear** transformation is to use this scheme successively for the stages that follow [9]. This is done by looking at the **Gray** coded input of the previous SNN as $\mathbf{b}_1 \mathbf{b}_2 \dots \mathbf{b}_n$ and then **take** the Gray code:of the Gray code.

Another possible technique for the **non-linear transformation** of the data is to use the real discrete Fourier transform (**RDFT**) [15]. The **RDFT** is a linear transform which can be made non-linear by truncating its output to 0 and 1 or -1 and 1. The **RDFT** can be computed using a fast transform which is known as **the** real fast discrete Fourier transform (**RFFT**) [16].

3.2.21 Weight Selection Schemes

The weights (α_j s) should reflect the goodness of the input data, **i.e.**, relatively large weights should be given to **input** data that can be classified with high accuracy. Various weight: selection **schemes** can be used to select weights for the **CNNs**. One possibility

is to use equal weights, which effectively takes the average of the outputs from the **SNNs**. Other possibilities include use of reliability measures which rank the sources according to their goodness. These reliability measures are, for example, source-specific **classification** accuracies of training data, source-specific overall **separabilities** of training data and equivocation **among** the data sources [1].

4. EXPERIMENTAL RESULTS

The **CNNs** were used to classify a data set **consisting** of the following 4 data sources:

- 1) **Landsat MSS** data (4 data channels)
- 2) **Elevation** data (in 10 m contour intervals, 1 data **channel**)
- 3) Slope data (0-90 degrees in 1 degree increments, 1 data **channel**).
- 4) Aspect data (1-180 degrees in 1 degree increments, 1 data **channel**)

Each channel comprised an **image** of 135 rows and 131 columns; all channels were co-registered.

The area used for classification was a **mountainous** area in Colorado having 10 ground-cover classes (Table 1). One class is water; the others are forest types. It is very difficult to distinguish among the forest types using the **Landsat** MSS data alone since the forest classes show very similar spectral response. In **addition**, as **seen** in Table 2. the **pairwise** JM distance separabilities [17] between most of the forest types in the **Landsat** MSS data are relatively low. **With** the help of elevation, slope and aspect data, the forest types can be better distinguished.

Reference data were compiled for the area by comparing a **cartographic** map to a color composite of the **Landsat** data and also to a line-printer output of each **Landsat** channel. By this method, 2019 **reference** points (11.4% of the area) were selected from two or more **homogeneous** fields in the imagery for each class. In the **first** **experiment** with the data, the largest field for each class **was** used as a training field and the other fields were used for testing the **classifiers**. Overall 1188 pixels were used for training and 831 pixels for testing the classifiers.

The CNN algorithms were implemented using one-layer conjugate-gradient delta rule neural networks [18],[19] as its SNNs. Using just one-layer SNNs makes each stage computationally less demanding. However, each stage can only guarantee **to** separate linearly separable data. The conjugate-gradient versions of the **feedforward** neural networks are computationally more efficient than conventional gradient descent neural networks [3],[18].

Table 1

**Training and Test Samples for Information Classes in
the First Experiment on the Colorado Data Set**

Class #	Information Class	Training Size	Test Size
1	Water	408	195
2	Colorado Blue Spruce	88	24
3	Mountane/Subalpine Meadow	45	42
4	Aspen	75	65
5	Ponderosa Pine	105	139
6	Ponderosa Pine/Douglas Fir	126	188
7	Engelmann Spruce	224	70
8	Douglas Fir/White Fir	32	44
9	Douglas Fir/Ponderosa Pine/Aspen	25	25
10	Douglas Fir/White Fir/Aspen	60	39
	Total	1188	831

Table 2

Pairwise JM Distances Between the 10
Information Classes in the Landsat MSS
Data Source [Maximum Separability is 1.00]

Class #	2	3	4	5	6	7	8	9	10
1	1.00	0.97	1.00	1.00	1.00	1.00	1.00	0.99	1.00
2	-	0.84	0.80	0.79	0.90	0.99	0.86	0.79	0.76
3	-	-	0.89	0.90	0.92	0.96	0.93	0.90	0.92
4	-	-	-	0.73	0.75	0.99	0.77	0.25	0.65
5	-	-	-	-	0.27	0.99	0.29	0.74	0.60
6	-	-	-	-	-	0.99	0.23	0.76	0.67
7	-	-	-	-	-	-	0.99	0.99	0.99
8	-	-	-	-	-	-	-	0.78	0.65
9	-	-	-	-	-	-	-	-	0.62
Average:	0.89								

The original input data were Gray coded and the non-linear transform for each succeeding stage was the Gray code of the preceding Gray code. Each SNN had 57 input neurons and 10 output neurons (one output neuron was set as 1 for each class the other neurons set equal to 0). In this experiment all the stages were given equal weights. For comparison the single-stage conjugate-gradient backpropagation (CGBP) algorithm with two layers (hidden, output) [18] was trained on the same data. All of the neural networks used the sigmoid activation function [9]. The CGBP neural network had 57 inputs, 32 hidden neurons and 10 output neurons. The experiment was run on a Gould NP-1 computer (as were all others).

The results of the first experiment are shown in Tables 3.a (training) and 3.b (test). The CNNS achieved its best results with 3 stages and 400 iterations per stage. The CNNP needed 4 stages and 300 iterations per stage. The best results with the CGBP were reached at 200 iterations. The training and classification time of the CNNP was the highest in this experiment. The reason for the time difference between CNNP and CNNS is that the CNNP needed 4 stages whereas the CNNS used only 3 stages.

Looking at the training results in Table 3.a, it is seen that the CGBP algorithm does a little better than the CNNS during training, both in terms of overall accuracy (OA) which is weighted by the number of pixels in each class and average (over the classes) accuracy (AVE). On the other hand, the test results in Table 3.b show that the CNNP with 4 stages is slightly better than the CGBP algorithm, in terms of overall and average accuracies for these data. The CNNP achieved around 0.7% better overall accuracy and about

Table 3

**Neural Network Methods Applied to Colorado Data.
First Experiment:
(a) Training Samples, (b) Test Samples.**

Table 3.a

Method	# of Its.	# of Stages	CPU Time	Percent Agreement with Reference for Class										OA	AVE
				1	2	3	4	5	6	7	8	9	10		
CNNS	400	3	1810	100.0	98.9	86.7	97.3	76.2	84.9	100.0	100.0	100.0	100.0	95.45	94.40
CNNP	300	4	2106	100.0	98.9	86.7	97.3	73.3	85.7	100.0	100.0	100.0	100.0	95.29	93.79
CGBP	200	1	1427	100.0	100.0	93.3	100.0	85.7	92.1	100.0	100.0	100.0	100.0	97.64	97.11
Number of Pixels				408	88	45	75	105	126	224	32	25	60	1188	1188

Table 3.b

Method	# of Its.	# of Stages	Percent Agreement with Reference for Class										OA	AVE
			1	2	3	4	5	6	7	8	9	10		
CNNS	400	3	99.5	83.3	50.0	49.2	10.1	43.1	100.0	4.5	8.0	82.1	56.32	52.98
CNNP	300	4	100.0	79.2	52.4	55.4	10.1	43.6	100.0	4.5	4.0	84.6	57.04	53.38
CGBP	200	1	99.0	83.3	45.2	38.5	17.3	40.4	100.0	2.3	12.0	94.9	56.32	53.29
Number of Pixels			195	24	42	65	139	188	70	44	25	39	831	831

0.1% better average accuracy for the test data. The test classification accuracies of the CNNs with 3 stages and CGBP were very similar.

The training data used in the experiment above were selected in such a way that one field for each class was used for training and the others as test data. It has been shown [3],[19] that neural networks are sensitive to having representative training samples. In order to see how well the CNNs compared to the CGBP with a more representative training sample, another experiment was conducted. In this experiment, training samples were selected uniformly spaced over the image. Approximately 50% of the samples were used for training and the rest for testing the neural networks (see Table 4).

The results of the second experiment are shown in Tables 5.a (training) and 5.b (test). Both CNNs used 3 stages and achieved their best results after 200 iterations per stage. The CGBP reached its best performance at 150 iterations. The CNNs with 3 stages needed about 250 CPU seconds more for training and classification than the CGBP. However, the CNNs are potentially much faster since they can be implemented in parallel stages. Looking at the training results in Table:5.a, it can be seen that all of the neural networks gave similar overall and average accuracies: i.e., with representative training samples, the training performance of all networks was almost the same. However, the test results in Table 5.b show that the three-stage CNNs outperformed the two-layer CGBP by more than 3.5%. It is also significant that these results are better than the best statistical results achieved in [3]. Therefore, the results are very

Table 4
 Training and Test Samples for Information Classes in
 the Second Experiment on the Colorado Data Set

Class #	Information Class	Training Size	Test Size
1	Water	301	302
2	Colorado Blue Spruce	56	56
3	Mountane/Subalpine Meadow	43	44
4	Aspen	70	70
5	Ponderosa Pine	157	157
6	Ponderosa Pine/Douglas Fir	122	122
7	Engelmann Spruce	147	147
8	Douglas Fir/White Fir	38	38
9	Douglas Fir/Ponderosa Pine/Aspen	25	25
10	Douglas Fir/White Fir/Aspen	49	50
	Total	1008	1011

Table 5

**Neural Network Methods Applied to Colorado Data.
Second Experiment:
(a) Training Samples, (b) Test Samples.**

Table 5.a

Method	# of Its.	# of Stages	CPU Time	Percent Agreement with Reference for Class										OA	AVE
				1	2	3	4	5	6	7	8	9	10		
CNNS	200	3	1190	100.0	85.7	74.4	91.4	68.9	78.7	100.0	50.0	76.0	98.0	86.81	79.79
CNNP	200	3	1190	100.0	83.9	74.4	91.4	68.2	78.7	100.0	47.4	76.0	98.0	87.10	81.80
CGBP	150	1	967	100.0	94.6	46.5	97.1	65.6	84.4	100.0	47.4	68.0	100.0	87.20	80.36
Number of Pixels				408	88	45	75	105	126	224	32	25	60	1188	1188

Table 5.b

Method	# of Its.	# of Stages	Percent Agreement with Reference for Class										OA	AVE
			1	2	3	4	5	6	7	8	9	10		
CNNS	200	3	100.0	80.4	50.0	87.1	59.9	78.7	99.3	31.6	44.0	90.0	82.49	72.10
CNNP	200	3	100.0	80.4	50.0	85.7	59.9	79.5	99.3	31.6	44.0	90.0	82.49	72.04
CGBP	150	1	100.0	92.9	36.4	67.1	57.3	70.5	98.6	28.9	36.0	80.0	78.93	66.77
Number of Pixels			195	24	42	65	139	188	70	44	25	39	831	831

satisfying, showing that the CNNs generalized well with representative training samples.

The results in both experiments showed that: the CNN architecture can be considered a desirable choice in multisource, **classification**, especially if training samples are representative. This architecture can also be used for other difficult classification problems. Although the CGBP algorithm showed, superior performance in training accuracy, it did not generalize as well as the CNN. These results were achieved by a network consisting of **one-layer** networks whereas the CGBP network is a **two-layer** network. As **noted** earlier, one-layer networks can only separate linearly separable data in contrast to the two-layer networks **which** can separate non-linearly separable data. Using **multilayer** stage networks in the CNN architecture is also a possibility, **but** it makes the training procedure computationally more complex.

5. CONCLUSIONS

Our experiments have shown the CNN architecture to be a **useful** alternative to conjugate-gradient backpropagation for **multisource** classification. Two versions of the architecture, the **CNNS** and the **CNNP**, were tested on a multisource data set **consisting of Landsat MSS** data, elevation data, slope data, and aspect data. The CNN algorithms outperformed the method of **conjugate-gradient** backpropagation in terms of test accuracy for this data set. The CNNs needed no more than 4 stages but more time-consuming in training **and** classification than the CGBP. However, they are

potentially much faster since they can be implemented in parallel stages.

At this point, the CNNs need to be tested more **extensively**. **Different** non-linear transformations and various weight-selection schemes need to be explored. Equal weights were used in the experiments reported here. Other weights could further improve the **accuracy** of the CNNs. The CNNs were trained on **binary** input data. Using continuous-valued inputs [19] for the CNNs is a subject of current research. Also, different types of CNN architectures are being explored. These architectures include CNNs with different non-linear transformations for each stage and different **numbers** of iterations for the stages.

REFERENCES

- [1] J.A. Benediktsson, P.H. Swain and O.K. Ersoy, "Neural Network Approaches Versus Statistical Methods in **Classification** of Multisource Remote Sensing Data," *IEEE Transactions on Geoscience and Remote Sensing*, vol. GE-28, no. 4, pp. 540-552, July 1990.
- [2] J.A. Benediktsson, P.H. Swain, O.K. Ersoy and D. Hong, "Classification of Very High Dimensional Data Using Neural Networks," *Proceedings IGARSS '90*, vol. 2, pp. 1269-1272, 1990.
- [3] J.A. Benediktsson and P.H. Swain, *Statistical Methods and Neural Network Approaches for Classification of Data from Multiple Sources*, Report No. **TR-EE 90-64**, Laboratory for Applications of Remote Sensing and School of Electrical Engineering, Purdue University, 1990.
- [4] J.A. Benediktsson and P.H. Swain, "Consensus Theoretic Classification Methods," submitted to *IEEE Transactions on Systems Man and Cybernetics*.
- [5] C. Genest and J.V. Zidek, "Combining Probability Distributions: A Critique and Annotated Bibliography," *Statistical Science*, vol. 1. no. 1, pp. 114-118, 1986.
- [6] K.J. McConway, "The Combination of Experts' Opinions in Probability Assessment: Some Theoretical Considerations," Ph.D. Thesis, University College, London 1980.
-

- [7] D. W. Ruck, S.K. Rogers, M. Kabrisky, M.E. Oxley, and B.W. Suter, "The Multilayer Perceptron as an Approximation to a Bayes Optimal Discrimination Function," *IEEE Transactions on Neural Networks*, vol. 1, no. 4, pp. 296-298, 1990.
- [8] T. Kohonen, "An Introduction to Neural Computing," *Neural Networks*, vol. 1, no. 1, pp. 3-16, 1988.
- [9] D. E. Rumelhart, G.E. Hinton and R.J. Williams, "Learning Internal Representation by Error Propagation," in *Parallel Distributed Processing: Explorations in the Microstructures of Cognition*, vol. 1, D.E. Rumelhart and J.L. McClelland (eds.), pp. 318-362, MIT Press, Cambridge, 1986.
- [10] O.K. Ersoy and D. Hong, "Parallel, Self-Organizing, Hierarchical Neural Networks," *IEEE Transactions on Neural Networks*, vol. 1, no. 2, pp. 167-178, 1990.
- [11] H. Valafar and O.K. Ersoy, *Parallel, Self-Organizing, Consensual Neural Network*, Report No. *TR-EE 90-56*, School of Electrical Engineering, Purdue University, 1990.
- [12] N. Nilsson, *Linear Machines*, McGraw-Hill, New York 1965.
- [13] L.K. Hansen and P. Salamon, "Neural Network Ensembles," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12, no. 10, pp. 993-1001, 1990.
-

[14] B.W. Silverman, Density Estimation for Statistics; and Data *Analysis*, Monographs on Statistics and Applied Probability, Chapman and Hall, New York **1986**.

[15] O.K. Ersoy, "Real Discrete Fourier Transform," IEEE *Transaction* on Acoustics, Speech and Signal Processing, **ASSP-33**, No. 4, pp. **880-882**, **1985**.

[16] N-C Hu and O.K. Ersoy, "Fast Computation of Real Discrete Fourier Transform for any Number of Data Points," to appear in IEEE *Transaction* on Circuits and Systems.

[17] P.H. Swain, "Fundamentals of Pattern Recognition in Remote Sensing", in Remote Sensing - *The Quantitative Approach*, edited by P.H. Swain and S.M. Davis, McGraw-Hill Book Company, New York, **1978**.

[18] E. Barnard and R.A. Cole, A Neural-Net Training *Program* Based on *Conjugate-Gradient* Optimization, Technical Report No. CSE 89-014, Oregon Graduate Center, July **1989**.

[19] J.A. Benediktsson, P.H. Swain and O.K. Ersoy, "Conjugate-Gradient Neural Networks in Classification of Multisource and Very-High Dimensional Remote Sensing Data," submitted to International Journal of Remote Sensing.