

The Guarded Neural Classifier

Torsten Felzer, Bernd Freisleben, and Martin Hoof

Abstract—This paper presents a new approach for classifying multi-dimensional data using an artificial neural network. The proposed network is an extension of a counterpropagation network with two special neurons controlling (or ‘guarding’) the ‘decisions’ of the network – that is why it is called Guarded Neural Classifier (GNC). The design of the GNC network is aimed at avoiding the most common drawbacks of neural network classifiers: the treatment of outliers in the training data and the misclassification of input patterns belonging to unknown classes during recall. The new approach is applied to the problem of partial discharge (PD) diagnosis, where reliable classification is of particular importance. It is shown that the network is able to recognize trained classes almost perfectly, and that the risk of misclassifications is considerably reduced in comparison to other solutions (for reliable training data, this risk even approaches zero).

Index Terms—Pattern classification, counterpropagation, vigilant counterpropagation, partial discharge diagnosis, outliers, rejection problem.

I. INTRODUCTION

IN many cases, an artificial neural network represents an implementation of a mapping from an n -dimensional input space onto an m -dimensional output space ($f : \mathbb{R}^n \rightarrow \mathbb{R}^m$). The reason why such neural networks have become more and more popular in the past decades is that the ‘mappings’ do not have to be ‘programmed’ like an ordinary computer program; they are ‘learned’ by the network in the course of a repetitive training process. Typically, parts of the mapping are known, and the network is trained by repeatedly presenting the known associations between input and desired output vectors – until the network has learned the underlying mapping up to a certain degree of perfection.

An outstanding feature is that the network then is not only able to recall the correct outputs for the input vectors memorized in the training phase, but also – in the ideal case – to produce reasonable outputs for unknown input patterns by generalizing from the known data. This generalization makes neural networks suitable for pattern recognition or classification tasks, since they can simply be trained by presenting several examples of each class and then – depending on whether or not the input space is represented adequately by the *training set* – the network should be able to classify a possibly infinite set of input vectors.

However, artificial neural networks used for classification often have two fundamental drawbacks. First, the training process is considerably disturbed if there is an ‘outlier’ in the training set, i.e. a pattern that deviates from the mapping induced by the other patterns, where the desired output simply does not

fit. Second, neural network classifiers often try to associate an input vector with an output class, even if the input vector is so ‘unknown’ that a classification into any one class does not make any sense, e.g. when the input belongs to a class that was not learned in the training phase. In this case, the network should be able to reject the input – which might also be called an *outlier* (in a wider sense) – as ‘unclassifiable’, or ‘unknown’.

Thus, the main goal in neural network based classification is to devise a *robust* network [3], [4], [5], [6], [7], which is able to handle outliers in the training phase *and* which rejects (or produces useful results for) outlier patterns in the recall phase. Such a network is confronted with two major tasks: the *detection* of outlier patterns (see e.g. [8], [9]) and the appropriate *treatment* of those patterns, which includes the *rejection* of the corresponding pattern [10] or the *removal* of associated points in feature space (see [11]).

Traditional backpropagation learning schemes [12] often result in unacceptable performance if the underlying training data contains outliers. The reason for this is that the usual backpropagation algorithm interpolates the data. Since outliers render interpolation useless, certain precautions have to be taken if there is *no guarantee* that the training data is *free of outliers* (e.g. [13], [14]).

In this paper, we present a neural network which does not suffer from the presence of outliers in both the training and recall data. It is based on the so-called FCPN (forward-only counterpropagation network) as introduced by Hecht-Nielsen [1], and the *vigilant* FCPN proposed previously by some of the authors and described in detail in [2]. The network involves two guards (G_t and G_r), observing the training phase and the recall phase, respectively, which is why it is called GNC network (Guarded Neural Classifier), and these two guards are responsible for possibly rejecting outliers during training and recall.

The proposed approach is applied to the problem of partial discharge (PD) diagnosis, where reliable classification is highly desirable for technical and economical reasons. It will be demonstrated that the network is able to recognize trained classes almost perfectly, and that the risk of misclassifications is considerably reduced in comparison to other solutions (for reliable training data, this risk even approaches zero).

The paper is organized as follows. Section II describes the architecture of the GNC network. In section III, the projected application field of partial discharge diagnosis is discussed. The results obtained in corresponding experiments are presented in section IV. Finally, section V concludes the paper and outlines areas for further research.

II. GNC NETWORK ARCHITECTURE

The general architecture of the GNC network is shown in figure 1.

T. Felzer and B. Freisleben are with the Department of Electrical Engineering and Computer Science (FB 12), University of Siegen, Hölderlinstr. 3, D-57068 Siegen, Germany.

M. Hoof is with ALSTOM (Switzerland) Inc. R&D Turbogenerators, Insulation Competence Center, CH-5242 Birr, Switzerland.

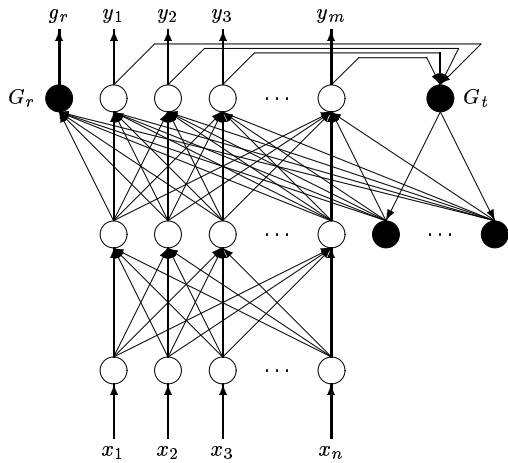


Fig. 1. The Guarded Neural Classifier

The network consists of three layers: an input layer (with size n), a classification layer (whose state is unknown or *hidden* to the user), and an output layer (with size m); the neurons in one layer are completely connected to the neurons in the next. A special role is played by the neurons drawn in black – their meaning will be explained later.

The connections between the layers are weighted and have to be trained prior to the ‘normal’ operation of the network. This is as follows: an input vector is presented to the network, and the classification neuron whose weight vector is closest to this input vector is found in a competitive process; this neuron (the *winner* of the competition) determines, via its weights to the output layer, the output of the network. Therefore, the classification layer consists of so-called *winner-take-all* neurons.

The training of the network consists of presenting a set of input/output vector pairs (the *training set*) to the network, and to adjust the connection weights such that, after several cycles through the training set, the network has ‘learned’ the associations therein, i.e. the network is able to compute (within certain limits) the output vector that belongs to any input vector of the training set. As for the weights between input and classification layer, the adjustment is done in an unsupervised, competitive manner [24], whereas for those between classification and output layer, it is done using a supervised learning technique [25].

The idea is that input vectors that are new to the network also produce an output which is based on the output produced by the nearest neighbor among the (input-to-hidden) weight vectors. Therefore, in the so-called *recall phase*, the network may be used as a kind of generalized lookup table observing the regularities ‘learned’ in the training phase, where the weight vectors have been organized to represent classes of input vectors.

More formally, the process of training the network and working with it may be described as follows. Let us denote the input vector as

$$\vec{x} = (x_1, x_2, x_3, \dots, x_n) \in \mathbb{R}^n.$$

The states of the neurons in the classification layer (excluding, for now, the neurons drawn in black) are represented by a vector

$$\vec{c} = (c_1, c_2, c_3, \dots, c_h),$$

with an appropriate dimension h , and the output of the network is a vector

$$\vec{y} = (y_1, y_2, y_3, \dots, y_m) \in \mathbb{R}^m.$$

The weights between the input and the classification layers may be denoted by a matrix

$$W = \begin{pmatrix} w_{11} & w_{12} & w_{13} & \dots & w_{1n} \\ w_{21} & w_{22} & w_{23} & \dots & w_{2n} \\ w_{31} & w_{32} & w_{33} & \dots & w_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ w_{h1} & w_{h2} & w_{h3} & \dots & w_{hn} \end{pmatrix} = \begin{pmatrix} \vec{w}_1 \\ \vec{w}_2 \\ \vec{w}_3 \\ \vdots \\ \vec{w}_h \end{pmatrix}$$

where w_{ij} stands for the weight value between classification neuron i and input neuron j and \vec{w}_i for the n -dimensional input-weight-vector of classification neuron i . Finally, the weights between the classification and the output layer are represented by a matrix

$$Q = \begin{pmatrix} q_{11} & q_{12} & q_{13} & \dots & q_{1h} \\ q_{21} & q_{22} & q_{23} & \dots & q_{2h} \\ q_{31} & q_{32} & q_{33} & \dots & q_{3h} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ q_{m1} & q_{m2} & q_{m3} & \dots & q_{mh} \end{pmatrix} = \begin{pmatrix} \vec{q}_1 \\ \vec{q}_2 \\ \vec{q}_3 \\ \vdots \\ \vec{q}_h \end{pmatrix}^T$$

with q_{li} being the weight between output neuron l and classification neuron i , whereas \vec{q}_i is the m -dimensional output-weight-vector of classification neuron i .

At the very beginning of the training phase, the components of the matrices W and Q are initialized with random values, and the input-weight-vectors \vec{w}_i ($i = 1, 2, 3, \dots, h$) are normalized to unit length in Euclidean space. Training the network now means associating input vectors with corresponding desired output vectors $\vec{d} = (d_1, d_2, d_3, \dots, d_m)$, i.e. the objective is to adjust the weights, so that after passing the input vector through the network, the output vector \vec{y} is closer to \vec{d} than before. This is done for every input/desired output vector pair (or for every pattern) in the training set.

Such a pass involves first determining the classification neuron whose input-weight-vector is closest to the input vector, which is, like the input-weight-vectors, also normalized to unit length. Because of this normalization, finding this classification neuron means to simply compute the dot products between \vec{x} and every input-weight-vector and taking the maximum. The dot product is also called the *activation* of the input-weight-vector by \vec{x} , with the simple geometrical interpretation of the cosine of the enclosed angle. In other words, the vector \vec{c} may be defined by

$$c_i = \begin{cases} 1 & \vec{x} \cdot \vec{w}_i \geq \vec{x} \cdot \vec{w}_j, j \neq i \\ 0 & \text{otherwise} \end{cases}. \quad (1)$$

The output of the network then is

$$\vec{y} := \vec{c} \times (Q^T), \quad (2)$$

where ‘ \times ’ denotes a matrix multiplication.

Thus, the classification neurons compete with each other to determine whose input-weight-vector is closest to the input vector – the dot product between this weight vector and the input vector is larger than that belonging to all other weight vectors, which means that the cosine of the enclosed angle is largest, so the distance between the two vectors is smallest. Equation (1) sets the output state of that neuron to 1. But since finding that neuron is done in a competitive process, it is important that only one of the c_i 's is set to 1, the other ones to 0. Thus, equation (1) has to be slightly modified, if more than one input-weight-vector has smallest distance to the input vector. In this case, the one with the smallest index wins by definition. However, in the following, equation (1) will characterize the winner – the reader should consider the additional rule without further mentioning.

Let i_0 be the index of the winner (i.e. $(c_{i_0}=1) \wedge (c_i=0, i \neq i_0)$), then equation (2) ensures that the output of the network is \vec{q}_{i_0} which means that the winner ‘has the right to’ determine the output.

In the training phase, every pass through the network is followed by an adaptation step, in which the weight matrices W and Q are adjusted in order to associate the inputs to the target outputs. Actually, only the weight vectors of the winner i_0 , i.e. \vec{w}_{i_0} and \vec{q}_{i_0} , are modified according to equations (3) and (4) at time t . All other weight vectors remain unchanged.

$$\vec{w}_{i_0}(t+1) = \vec{w}_{i_0}(t) + \alpha \cdot (\vec{x}(t) - \vec{w}_{i_0}(t)) \quad (3)$$

$$\vec{q}_{i_0}(t+1) = \vec{q}_{i_0}(t) + \beta \cdot (\vec{d}(t) - \vec{q}_{i_0}(t)) \quad (4)$$

$$\alpha(e+1) = \frac{\alpha(e)}{1+\gamma}, \quad \beta(e+1) = \frac{\beta(e)}{1+\xi} \quad (5)$$

Equation (3) represents the competitive learning rule of [24], with the simple interpretation that the input-weight-vector \vec{w}_{i_0} is moved in the direction of the input vector \vec{x} . The parameter $\alpha > 0$ determines the magnitude of this ‘movement’, and after one cycle (also called an *epoch*) e through the entire training set, this parameter is decreased (equation (5), with $\gamma > 0$) before the next cycle is performed. Consequently, the changes become smaller and smaller, which should be in accordance with the network’s degree of ‘expertise’.

As already mentioned, initially \vec{x} as well as the input-weight-vectors are normalized to unit length, but this does, in general, *not* hold for \vec{w}_{i_0} after the adaptation step. Since the input-weight-vectors always stay within the unit sphere, this minor deviation can be neglected though, in other words, \vec{c} can still be computed as if the \vec{w}_i 's were normalized.

The adjustment of the output-weight-vectors according to (4), which is based on Grossberg’s outstar structure [25], may be interpreted correspondingly: the output-weight-vectors are moved in the direction of the target output vector, and the magnitude of this movement is governed by the parameter $\beta > 0$ which is also decreased over time (see equation (5), with $\xi > 0$).

As stated above, the adjustment of the weights is performed repetitively for several epochs through the training set, in this way ‘recognizing’ clusters of ‘similar’ input vectors and ‘learning’ the mean target output vectors associated with the inputs in

each cluster. One representative of the clusters (or *classes*) will be encoded in each input-weight-vector, and the mean output of all vectors in a cluster will be encoded in the corresponding output-weight-vector. Training is stopped either after a preset number of epochs or after the total error, i.e. the (squared) difference between the target output vector and the actual output vector summed over all patterns in the training set, drops below a certain threshold, which means that a certain convergence has been reached.

The network defined so far, which is merely a standard FCPN, has two fundamental drawbacks. First, two ‘similar’ input vectors will always activate the same ‘winning’ classification neuron, provided that the ‘similarity’ is strong enough. Therefore, those similar inputs will produce the same output, especially in the training phase. This behavior is perfectly desirable as long as the two target output vectors are not too different. But if one of the patterns is an outlier, it does not comply with the mapping induced by the other pairs in the training set. Its input vector seems to belong to a class of ‘similar’ vectors, but the target output vector is extremely dissimilar to all target output vectors of the other patterns in the mentioned class. In this case, the training process is disturbed considerably, because equation (4) will move the output-weight-vector in totally different directions in every epoch. Consequently, it is highly unlikely that convergence will ever be found.

The second problem of the standard FCPN relates to the way input vectors from formerly unknown classes are treated. Even if an input vector in the recall phase is very far away (in Euclidean space) from all those ‘seen’ in the training phase and thus from all input-weight-vectors, there is always (at least) one input-weight-vector which is closest to that input. The network then takes the output-weight-vector of the neuron winning the competition, and returns it as overall output, regardless of how unknown the input really was.

To avoid these two problems, two guards (G_t and G_r), observing the training phase and the recall phase, respectively, are introduced, together with a number of initially unused classification neurons. The guards as well as the extra classification neurons are those drawn in black in figure 1.

The G_t neuron takes care of the outliers in the training phase. Each time an outlier is detected, a new, formerly unused classification neuron is activated, whose weight vectors are assigned to this outlying pattern, i.e. the outlier ‘gets’ its ‘own’ class, and thus does not disturb the training of the other patterns any more. In the recall phase, the G_r neuron examines the ‘degree of similarity’ between the input vector and the winning classification neuron, and compares it to all those experienced during the training phase. If this similarity is too low, then the guard G_r assumes the input to be unknown and reports this as an additional output to the user. In other words, G_r indicates whenever the output of the network is not to be relied upon, i.e. the classification is uncertain.

For detecting an outlier, the distance between the actual and the target output vectors has to be examined, which leads to computing the error E :

$$E = \sum_{i=1}^m (d_i - y_i)^2. \quad (6)$$

If this error is above a predefined threshold, then the G_t neuron ‘concludes’ that the currently examined pattern is an outlier. In this case, the training has to deviate from equations (3) and (4). The modified training simply consists of taking a formerly unused classification neuron from the pool already mentioned above and setting its weight vectors appropriately. Let the index of this additional neuron be $h+g$, then this simply amounts to applying equations (7) and (8):

$$\vec{w}_{h+g}(t+1) = \vec{x}(t) \quad (7)$$

$$\vec{q}_{h+g}(t+1) = \vec{d}(t) \quad (8)$$

From this time on, the corresponding classification neuron takes part in the competition. Equation (7) ensures that the activation value will be

$$\vec{w}_{h+g} \cdot \vec{x} = 1.0,$$

when the outlier in question is presented to the network during the next epoch – the vectors \vec{x} and \vec{w}_{h+g} are both equal and normalized to unit length. Since the value 1.0 is the maximum possible activation value (the maximum of the cosine function), the new classification neuron is guaranteed to win next time. Therefore, the outlier does not decrease the overall training performance in any way.

Due to the random initialization of the matrix Q , the error E will obviously be relatively large during the first few epochs, which would mean that nearly all patterns are classified by G_t as outliers. Consequently, the operation of G_t is not started until an appropriate number of epochs is completed. Thus, the network has time to self-organize its classification.

The information on which G_r bases its decision whether or not a pattern is unknown to the network is investigated during an additional cycle through all the patterns in the training set T at the end of the training phase (without any weight adaptation). This is done to find the minimum activation values θ_i for the winning classification neurons ($i = 1, 2, 3, \dots, H$, where $H \geq h$ is the total number of classification neurons, including those ‘added’ during the training phase):

$$\theta_i = \min_{\vec{x}_t \in T} \{ \vec{x}_t \cdot \vec{w}_i \mid \vec{x}_t \cdot \vec{w}_i \geq \vec{x}_t \cdot \vec{w}_j, j \neq i \} \quad (9)$$

This minimum activation value, multiplied by some tolerance factor $0 < \tau \leq 1$, is used as a threshold to decide on the usefulness of the network output. Whenever a classification neuron i_0 wins with an activation value *less than or equal to* $\tau \cdot \theta_{i_0}$, then G_r sets the additional output g_r to 1:

$$g_r = \begin{cases} 1 & (\vec{x}_r \cdot \vec{w}_{i_0} \geq \vec{x}_r \cdot \vec{w}_i, i \neq i_0) \\ & \wedge (\vec{x}_r \cdot \vec{w}_{i_0} \leq \tau \cdot \theta_{i_0}) \\ 0 & \text{otherwise} \end{cases}, \quad (10)$$

where \vec{x}_r is the input vector presented to the network.

If a classification neuron wins that has never won during the additional cycle through the training set, the output \vec{y} of the network obviously does not make too much sense. Therefore, g_r should also be set to 1 in this case, which can be easily accomplished by setting the minimum activation value θ_{i_1} for such an

‘unidentified’ classification neuron i_1 to $1/\tau$ before applying equation (10).

Since the computation of the output vector \vec{y} remains unchanged (compared to the standard FCPN), the user still has the ‘unguarded’ network output, but is additionally informed about its usefulness by the binary output value g_r .

III. APPLICATION: PARTIAL DISCHARGE ANALYSIS

To demonstrate the performance of the neural network approach presented in section II, it is applied to insulation diagnostics using the analysis of partial discharges (PD), which is a special field in electrical power engineering. A PD is a locally confined electrical breakdown in the insulation system of any high voltage equipment like e.g. transformers, generators, high voltage motors or power cables. Those partial breakdowns may occur rather frequently during the operation of the equipment, and they may precede a forthcoming failure of the device.

There are several different insulation defect categories describing the source of the PD – some are more, others are less severe. The goal of partial discharge diagnosis is now to identify the defect category of observed PD patterns, thus determining when to repair or replace critical equipment, in order to avoid expenses due to unscheduled down times of the device in question.

This paper distinguishes between six defect categories, which are called, for simplicity reasons, DC_1, DC_2, \dots, DC_6 . These categories are motivated and investigated with respect to their physical meaning in [26], which treats the topic of partial discharges in detail.

Extracting the input feature vectors defining a specific PD pattern is dependent on so-called parameter vectors. Those vectors, here simply called P_1, P_2, \dots, P_7 , influence the discriminating content among the input data. The exact meaning and definition of the used parameters is beyond the scope of this paper, and the interested reader is again referred to [26]. However, it is not necessary to fully understand the theory of partial discharge diagnosis in order to evaluate the performance of the GNC network in the application following below.

IV. RESULTS

In order to apply the GNC network to the problem introduced in the last section, 260 partial discharge patterns resulting from known defect categories have been measured in different insulation system environments. This data has been divided into two sets: the training set (already explained in section II) and the test set used for evaluating the performance of the network.

Each measurement results in seven input/output pairs (for each of the seven parameter vectors), consisting of 64-dimensional, normalized input vectors and 6-dimensional target output vectors. The exact partitioning (training/test vectors) for each of the six defect categories is given in table I.

The encoding of the six defect categories in the 6-dimensional output vectors consists of five 0-valued components and one component with value 1.0 (representing the corresponding defect category).

TABLE I
NUMBER OF TRAINING AND TEST VECTORS

Defect Category	Number of Training Vectors	Number of Test Vectors
DC_1	15	25
DC_2	20	25
DC_3	20	25
DC_4	20	25
DC_5	20	25
DC_6	15	25
TOTAL	110	150

In the recall phase, an actual output vector merely has to satisfy a much weaker condition in order to represent a particular defect category k :

$$(y_k \geq 0.9) \wedge (y_l \leq 0.1, l \neq k). \quad (11)$$

To demonstrate the effects of both guards G_t and G_r , two kinds of experiments have been conducted. In the first experiment, all 110 training vectors, i.e. vectors representing *all* six defect categories, have been presented to the network, and the network's recognition performance has been evaluated using the 150 test vectors. The network parameters in this experiment, which were found empirically (except for $n = 64$ and $m = 6$), were as follows:

- $h = 30$ classification neurons were used initially;
- the components of the input-weight-vectors \vec{w}_i were initialized with random values between 0.0 and 1.0, and the \vec{w}_i were subsequently normalized to unit length (as already mentioned in section II);
- the components of the output-weight-vectors \vec{q}_i were initialized with random values between 0.0 and 0.1 (without any normalization), which makes sure that condition (11) is *not* satisfied, when the winner in the recall phase has *never* won in the entire training phase;
- the 'learning parameters' were initialized with $\alpha = 0.6$ and $\beta = 0.1$ and modified using $\gamma = \xi = 0.2$ (see equation (5)) after each epoch;
- the training guard G_t was 'activated' after five epochs, and it classified a pattern as an 'outlier' if the error E was above 0.5;
- the tolerance factor τ (see equation (10)) was empirically chosen to be $\tau = 0.65$.

As already mentioned above, it is not really necessary to know the exact meaning of the parameter vectors $P_1 - P_7$ to understand the results of this experiment. However, it *is* necessary to know that the type of parameter vector determines the discriminating potential (as far as PD classification is concerned) of the input data. It should be noted that P_1 leads to the least discriminating data, $P_2 - P_5$ produce moderate results, and P_6 and P_7 lead to the most reliable input data, regarding classification of individual classes. These empirical findings are directly reflected in the number of extra classification neurons 'opened' by G_t in the training phase (see table II): the more discriminating the data, the less extra classification neurons are needed.

The results obtained when testing the trained GNC with the 150 test vectors are presented in table III. It can be seen that the

TABLE II
NUMBER OF EXTRA CLASSIFICATION NEURONS

Training Phase (GNC)	Parameter Vector						
	P_1	P_2	P_3	P_4	P_5	P_6	P_7
# of 'Outliers'	13	3	5	5	2	0	0

TABLE III
SUCCESS RATES OF THE GNC NETWORK (IN %)

Defect Category	Parameter Vector						
	P_1	P_2	P_3	P_4	P_5	P_6	P_7
DC_1	100	100	88	100	100	100	100
DC_2	96	96	100	100	96	100	100
DC_3	96	100	100	100	96	100	100
DC_4	88	100	100	100	96	100	100
DC_5	100	96	92	100	100	100	100
DC_6	100	100	100	100	100	100	100
Overall Success	96.67	98.67	96.67	100	98	100	100

performance is almost perfect, even for the parameter vector P_1 .

To be able to evaluate this result, the experiment has also been performed using the standard FCPN. The parameters were chosen exactly as in the GNC case, though the FCPN has no guards and also no extra classification neurons. The corresponding results are listed in table IV, and the striking characteristic is that the FCPN performs much worse, except for 'nice' data.

A graphical representation of the result in this first recognition experiment is depicted in figure 2.

In a second experiment, only training vectors belonging to one or a combination of two defect categories were trained in 21 different training sets (the GNC network was reinitialized and retrained for every single training set). However, the network was tested each time using all 150 test vectors, so many test vectors belonged to 'unknown' categories and should thus be rejected.

This experiment was conducted with three different network types: a GNC network without recall guard, a full GNC network, and a backpropagation network [27], an architecture that is often used in PD analysis [15], [16], [17], [18], [19], [20], [21], [22], [23]. The parameters for the GNC network did not differ from those in the recognition experiment above – except for $\tau = 0.85$, which accounts for the objective to reliably reject 'unknown input vectors'. The rejection of a vector is either

TABLE IV
SUCCESS RATES OF THE FCPN (IN %)

Defect Category	Parameter Vector						
	P_1	P_2	P_3	P_4	P_5	P_6	P_7
DC_1	100	100	88	100	100	100	100
DC_2	0	96	96	100	92	100	100
DC_3	0	0	0	0	96	100	100
DC_4	0	100	100	100	92	100	100
DC_5	100	96	92	100	100	100	100
DC_6	100	0	0	0	100	100	100
Overall Success	50	65.33	62.67	66.67	96.67	100	100

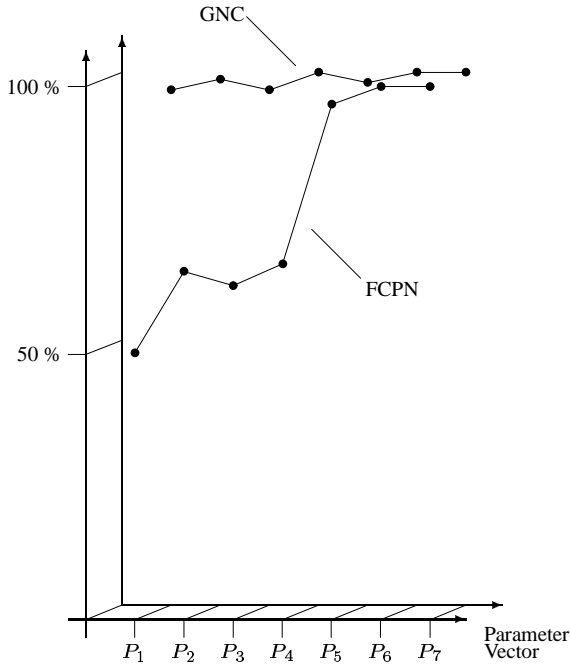


Fig. 2. Recognition Performance

TABLE V
CORRECTLY REJECTED TEST VECTORS (IN %)

Network Architecture	Parameter Vector						
	P ₁	P ₂	P ₃	P ₄	P ₅	P ₆	P ₇
no G_r neuron	59.8	42.9	57.7	43.8	68.5	75.2	74.4
Full GNC	78.7	76.8	82.5	82.2	98.6	100	99.2
Backprop Net	10.5	13.9	15.5	15.8	18.6	25.7	20.5

symbolized by a ‘firing’ G_r neuron or by the fact that condition (11) is not satisfied. A detailed description of the architecture and the training process of the backpropagation network can be found in [12].

In table V, it can be seen that the GNC network without G_r does not perform too badly (in comparison to the backpropagation network), but not quite as good as the full GNC network, which performs almost perfectly for “high quality” input data. The superiority of the GNC becomes obvious when the catastrophic result of the backpropagation network is considered. As already mentioned, backpropagation networks cannot cope with outliers in classification tasks due to their interpolative nature.

The result is displayed graphically in figure 3.

V. CONCLUSIONS

An approach for the automated classification of multi-dimensional real-valued data using an artificial neural network has been presented. The network – which has been named GNC (for Guarded Neural Classifier) – is an extension of a forward-only counterpropagation network (FCPN). It has its name from two additional neurons, the ‘guards’, observing the training and recall phases. The specific task of those guards is to eliminate two well-known problems associated with neural network classifiers: first, the treatment of outliers in the training phase, and

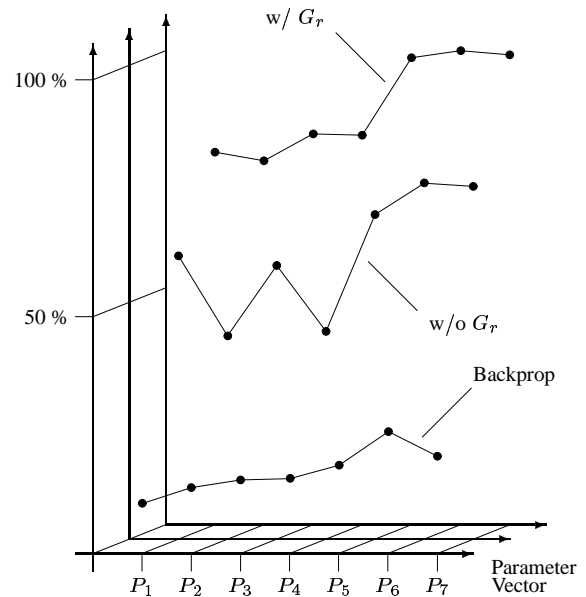


Fig. 3. Rejection Performance

second, the rejection of input vectors belonging to unknown classes in the recall phase.

The GNC network has been applied to the problem of partial discharge diagnosis, with the results compared to the ‘un-guarded’ version of the net. It can be concluded that the GNC is able to recognize patterns very reliably, even if the discriminating content of the training data is quite low, which usually makes it very hard for neural network classifiers to perform well. Moreover, the performance of the GNC in the rejection problem, where unknown test vectors are expected to be rejected as ‘unclassifiable’, is very impressive. The network performance is nearly perfect when the degree of discrimination in the training data is sufficiently large.

There are several areas for future research. First, in order to further evaluate the GNC network, it would be interesting to apply the network to a whole range of various problems and to compare the results with those obtained with the help of several other classifiers.

Second, an issue that also should be dealt with in the near future is the way the output-weight-vectors are trained. It is questionable whether it is really necessary to change them incrementally into the direction of the target output vector, when the output encoding scheme of section IV (“1-of- m ” encoding) is used. Instead, setting the weight to the target output vector might be adequate (this has to be examined).

Finally, the recall guard of the GNC outputs an additional binary value indicating whether or not the network output is based on a reliable decision. Alternatively, it might be desirable to have values between 0 and 1, estimating, for each output class, the probability the current input pattern belongs to that class.

REFERENCES

- [1] R. Hecht-Nielsen, “Counterpropagation networks,” *Applied Optics*, vol. 26, pp. 4979–4984, 1987.

- [2] B. Freisleben, M. Hoof, and R. Patsch, "Using counterpropagation neural networks for partial discharge diagnosis," *Neural Computing & Applications*, pp. 318–333, July 1998.
- [3] S.-R. Lay and J.-N. Hwang, "Robust construction of radial basis function networks for classification," in *Proceedings of the IEEE International Conference on Neural Networks*, 1993, vol. 3, pp. 1859–1864.
- [4] K. Liano, "A robust approach to supervised learning in neural networks," in *Proceedings of the IEEE International Conference on Neural Networks*, 1994, vol. 1, pp. 513–516.
- [5] K. Liano, "Robust error measure for supervised neural network learning with outliers," *IEEE Transactions on Neural Networks*, vol. 7, no. 1, pp. 246–250, 1996.
- [6] J. Larsen, L. Nonboe, M. Hintz-Madsen, and L. K. Hansen, "Design of robust neural network classifiers," in *Proceedings of the 1998 IEEE International Conference on Acoustics, Speech and Signal Processing*, 1998, vol. 2, pp. 1205–1208.
- [7] J.T. Lo and D. Bassu, "Training multilayer perceptrons in the presence of measurement outliers," in *Proceedings of the 2001 International Joint Conference on Neural Networks*, Vol. 3, 2001, pp. 2030–2035.
- [8] C. Zhang, P. M. Wong, and O. Selinus, "A comparison of outlier detection methods: exemplified with an environmental geochemical dataset," in *Proceedings of the 6th International Conference on Neural Information Processing*, 1999, vol. 1, pp. 183–187.
- [9] C. López, "Looking inside the ANN "black box": classifying individual neurons as outlier detectors," in *Proceedings of the International Joint Conference on Neural Networks*, 1999, vol. 2, pp. 1185–1188.
- [10] J. Liu and P. Gader, "Outlier rejection with MLPs and variants of RBF networks," in *Proceedings of the 15th International Conference on Pattern Recognition*, 2000, vol. 2, pp. 680–683.
- [11] P. L. Rosin and F. Fierens, "Improving neural network generalisation," in *Proceedings of the International Geoscience and Remote Sensing Symposium*, 1995, vol. 2, pp. 1255–1257.
- [12] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," in *Parallel Distributed Processing: Explorations in the Microstructures of Cognition*, D. E. Rumelhart and J. L. McClelland, Eds., Cambridge, MA, 1986, vol. 1, pp. 318–362, MIT Press.
- [13] C. L. P. Chen, "A rapid supervised learning neural network for function interpolation and approximation," *IEEE Transactions on Neural Networks*, vol. 7, no. 5, pp. 1220–1230, 1996.
- [14] C.-C. Chuang, S.-F. Su, and C.-C. Hsiao, "The annealing robust back-propagation (ARBP) learning algorithm," *IEEE Transactions on Neural Networks*, vol. 11, no. 5, pp. 1067–1077, 2000.
- [15] N. Hozumi, T. Okamoto, and T. Imajo, "Discrimination of partial discharge patterns using a neural network," *IEEE Transactions on Electrical Insulation*, vol. 27, no. 3, pp. 550–556, 1992.
- [16] L. Satish and B. I. Gururaj, "Partial discharge pattern classification using multilayer neural networks," *IEE Proceedings-A*, vol. 140, no. 4, pp. 323–330, 1993.
- [17] A. A. Mazroua, M. A. A. Salama, and R. Bartnikas, "PD pattern recognition with neural networks using the multilayer perceptron technique," *IEEE Transactions on Electrical Insulation*, vol. 28, no. 6, pp. 1082–1089, 1993.
- [18] L. Satish and W. S. Zaengl, "Artificial neural networks for recognition of 3-D partial discharge patterns," *IEEE Transactions on Dielectrics and Electrical Insulation*, vol. 1, no. 2, pp. 265–275, 1994.
- [19] M. Oyama, E. Hanai, H. Aoyagi, H. Murase, I. Ohshima, and S. Menju, "Development of detection and diagnostic techniques for partial discharge in GIS," *IEEE Transactions on Power Delivery*, vol. 9, no. 2, pp. 811–818, 1994.
- [20] A. A. Mazroua, R. Bartnikas, and M. M. A. Salama, "Discrimination between PD pulse shapes using different neural network paradigms," *IEEE Transactions on Dielectrics and Electrical Insulation*, vol. 1, no. 6, pp. 1119–1131, 1994.
- [21] T. Okamoto and T. Tanaka, "Partial discharge pattern recognition for three kinds of model electrodes with a neural network," *IEE Proc.-Sci. Meas. Technol.*, vol. 142, no. 1, pp. 75–84, 1995.
- [22] C. Cachin and H. J. Wiesmann, "PD recognition with knowledge-based preprocessing and neural networks," *IEEE Transactions on Dielectrics and Electrical Insulation*, vol. 2, no. 4, pp. 578–589, 1995.
- [23] A. Krivda, *Recognition of discharges discrimination and classification*, PhD-Thesis, Delft University of Technology, The Netherlands, 1995.
- [24] T. Kohonen, *Self-organization and associative memory*, Springer-Verlag, 1989.
- [25] S. Grossberg, "Embedding fields: A theory of learning with physiological implications," *Mathematical Psychology*, vol. 6, pp. 209–239, 1969.
- [26] M. Hoof, *Pulse-sequence-analysis: A new method of partial discharge diagnosis (in German)*, PhD Thesis, Universität-GH Siegen, Germany, 1997.
- [27] D. E. Rumelhart and J. L. McClelland, Eds., *Parallel distributed processing: explorations in the microstructures of cognition*, MIT Press, Cambridge, MA, 1986.