

## The Guarded Neural Classifier

Torsten Felzer, Bernd Freisleben, and Martin Hoof

**Abstract**—A new classification approach aimed at eliminating problems with outliers in the training data and unknown inputs at recall is presented. The proposed network is an extended counterpropagation network with mechanisms to guard its decisions. To show the feasibility of the approach, it is applied to partial discharge (PD) diagnosis.

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

### I. INTRODUCTION

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’.

A *robust* network (e.g. [1]), which is able to handle these problems, is confronted with two major tasks: the detection of outlier patterns (see e.g. [2]) and the appropriate treatment of those patterns, which includes the rejection of the corresponding pattern [3] or the removal of associated points in feature space (see [4]).

Traditional backpropagation learning schemes [5] often result in unacceptable performance if the underlying training data contains outliers, because the usual backpropagation algorithm interpolates the data. But outliers render interpolation useless, so certain precautions have to be taken if the training data could contain outliers (e.g. [6]).

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 [7], and the *vigilant* FCPN described in detail in [8]. The network involves two guards ( $G_i$  and  $G_r$ ), taking care of outliers in the training and recall phases, respectively, which is why it is called GNC network (Guarded Neural Classifier).

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 (of input patterns belonging to untrained classes) is considerably reduced in comparison to other solutions.

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 and the results obtained in corresponding experiments are presented. Finally, section IV concludes the paper.

### II. GNC NETWORK ARCHITECTURE

As shown in figure 1, the architecture of the GNC network combines the FCPN (explained in the following) with a number of additional neurons drawn in black (to be explained later).

T. Felzer is with the Department of Computer Science, Darmstadt University of Technology, Alexanderstr. 10, D-64283 Darmstadt, Germany.

B. Freisleben is with the Department of Mathematics and Computer Science,

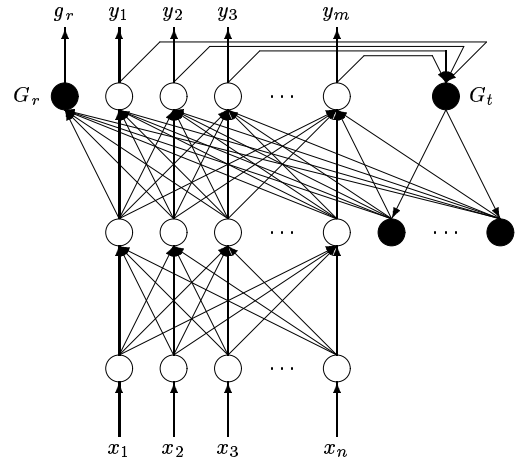


Fig. 1. The Guarded Neural Classifier

The network (excluding, for now, the neurons drawn in black) consists of three completely-connected layers: an input layer (with size  $n$ ), a (hidden) classification layer (with an appropriate size  $h$ ), and an output layer (with size  $m$ ). The connection weights may be denoted by two matrices  $W$  and  $Q$ :

$$W \stackrel{def}{=} (w_{ij}) \stackrel{def}{=} \begin{pmatrix} \vec{w}_1 \\ \vec{w}_2 \\ \vdots \\ \vec{w}_h \end{pmatrix}, Q \stackrel{def}{=} (q_{ki}) \stackrel{def}{=} \begin{pmatrix} \vec{q}_1 \\ \vec{q}_2 \\ \vdots \\ \vec{q}_m \end{pmatrix}^T$$

where  $w_{ij}$  stands for the weight value between classification neuron  $i$  and input neuron  $j$ ,  $\vec{w}_i$  for the  $n$ -dimensional *input-weight-vector* of classification neuron  $i$ ,  $q_{ki}$  for the weight between output neuron  $k$  and classification neuron  $i$ , and finally  $\vec{q}_i$  for the  $m$ -dimensional *output-weight-vector* of classification neuron  $i$ .

The weights define a mapping from the  $n$ -dimensional input vector  $\vec{x}$  onto the  $m$ -dimensional output vector  $\vec{y}$  which works as follows: the classification neuron whose input-weight-vector is closest to  $\vec{x}$  is found in a competitive process; this neuron (the *winner* of the competition) determines  $\vec{y}$  via its output-weight-vector.

Before this mapping actually produces useful results, the weights have to be trained in an initial training phase by iteratively presenting pairs of input and corresponding desired output vectors contained in the *training set*. At the very beginning, the components of the matrices  $W$  and  $Q$  are initialized with (small) random values and adjusted after each presentation.

To facilitate the competitive process mentioned above, it is made sure that the input vectors as well as the input-weight-vectors  $\vec{w}_i$  have unit length by normalizing them. As a consequence, the dot product of those vectors – denoted as activation  $a_i$  of classification unit  $i$  – is equal to the cosine of the enclosed angle, and  $a_i$  is largest, when the enclosed angle is smallest, i.e. when the two vectors are closest.

Therefore, the maximum component of the  $h$ -dimensional vector  $\vec{a} \stackrel{def}{=} W \times \vec{x}$ , where  $\times$  denotes matrix multiplication, corresponds to the winner of the competition. If we define a 1-of- $h$ -vector (i.e. an  $h$ -dimensional vector with exactly one component equal to 1.0 and all

University of Marburg, Hans-Meerwein-Str., D-35032 Marburg, Germany.

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

others equal to 0.0)  $\vec{c}$  by

$$c_i \stackrel{def}{=} \begin{cases} 1 & \vec{x} \cdot \vec{w}_i = a_i \geq a_j, \forall j \neq i \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

(with the additional rule that only the component with the smallest index is set to 1.0, if multiple classification neurons possess maximum activation), then the output of the network may simply be computed as

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

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  $\vec{q}_{i_0}$ .

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 desired 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) – all other weight vectors remain unchanged:

$$\vec{w}_{i_0}^{new} = \vec{w}_{i_0}^{old} + \alpha \cdot (\vec{x} - \vec{w}_{i_0}^{old}), \quad (3)$$

$$\vec{q}_{i_0}^{new} = \vec{q}_{i_0}^{old} + \beta \cdot (\vec{d} - \vec{q}_{i_0}^{old}), \quad (4)$$

where  $\vec{d}$  denotes the desired output belonging to the input vector  $\vec{x}$ .

Equation (3) represents the (unsupervised) competitive learning rule of [9], 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 adjustment of the output-weight-vectors according to (4), which is based on Grossberg's outstar structure [10], may be interpreted analogously (although it employs *supervised* learning): the output-weight-vectors are moved in the direction of the desired output vector.

The learning rates  $\alpha > 0$  and  $\beta > 0$  – which determine the respective magnitudes of these 'movements' – are decreased after each cycle through the entire training set (also called an *epoch*) according to equation (5) (with  $\gamma > 0$  and  $\xi > 0$ ):

$$\alpha^{new} = \frac{\alpha^{old}}{1 + \gamma}, \quad \beta^{new} = \frac{\beta^{old}}{1 + \xi}. \quad (5)$$

Consequently, the changes become smaller and smaller, which should reflect the network's degree of 'expertise'.

As already mentioned, initially the input-weight-vectors are (like  $\vec{x}$ ) 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 (and thus cannot diverge), there is no need for 're-normalization', though. Training is stopped either after a preset number of epochs or after the total error summed over all patterns in the training set has dropped below a certain threshold.

The network defined so far has two fundamental drawbacks. First, 'similar' input vectors will always activate the same 'winning' classification neuron. Applying equation (4) therefore causes problems, if one of the corresponding patterns is an outlier, since the desired output vector of that pattern is extremely dissimilar to all desired output vectors of the other patterns in the class of similar inputs represented by the aforementioned classification neuron. As a consequence, that neuron's output-weight-vector would be 'moved' in totally different directions within the same epoch. The second drawback 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 from all those 'seen' in the training phase, the network still returns the output-weight-vector of the neuron with maximum (though relatively small) activation.

To avoid these problems, two guards ( $G_t$  and  $G_r$ ), responsible for training and recall phases, 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 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 process 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, it assumes the input to be unknown and reports this as an additional output to the user, so  $G_r$  indicates whether or not the network's output is to be relied upon.

For detecting an outlier, the error  $E$  between actual and desired output has to be computed:

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

If this error is above a threshold  $E_{max}$ , 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 and setting its weight vectors appropriately. Let the index of this additional neuron be  $h + e$ , then this simply amounts to applying equation (7):

$$\vec{w}_{h+e} \stackrel{def}{=} \vec{x}, \quad \vec{q}_{h+e} \stackrel{def}{=} \vec{d}. \quad (7)$$

From this time on, the corresponding classification neuron takes part in the competition, and equation (7) ensures that it will become the 'winner', when the outlier in question is presented to the network during the next epoch (because  $a_{h+e}$  will then be 1.0, which is the maximum possible activation value).

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 a small number  $e_0$  of epochs has been completed.

$G_r$  bases its decision whether or not a pattern is unknown to the network on  $H$  values (where  $H \geq h$  is the total number of classification neurons, including those 'added' during the training phase) – denoted  $\lambda_i$ , one for each classification neuron ( $i = 1, 2, 3, \dots, H$ ) – computed during a final (additional) cycle through all the patterns in the training set  $T$  at the end of the training phase (without any weight adaptation).  $\lambda_{i_0}$  is the minimum activation value among all those corresponding to the training pairs that caused classification neuron  $i_0$  to become the winner. Whenever a classification neuron  $i_0$  wins with an activation value below the threshold  $\tau \cdot \lambda_{i_0}$  (with  $0 < \tau \leq 1$ ), then  $G_r$  sets the additional (binary) output  $g_r$  to 1:

$$g_r \stackrel{def}{=} \begin{cases} 1 & \vec{x}_r \cdot \vec{w}_{i_0} \leq \tau \cdot \lambda_{i_0} \\ 0 & \text{otherwise} \end{cases}, \quad (8)$$

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

The  $\lambda$ -value of a classification neuron that has never won during the final training cycle is set to  $1/\tau$  which makes sure that  $G_r$  *always* 'complains', if such an 'unidentified' classification neuron wins in the recall phase.

### III. APPLICATION AND RESULTS

To demonstrate the performance of the GNC network, 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

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                    |

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 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 [11], 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 [11]. 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.

In order to apply the GNC network to the PD problem, 260 partial discharge patterns resulting from known defect categories have been measured in different insulation system environments. This data has been divided into the training set and the test set (the partitioning with respect to each of the categories is given in table I). Each of the 260 measurements results in seven input/output pairs (for each of the seven parameter vectors), consisting of 64-dimensional, normalized input vectors and 6-dimensional desired output vectors.

The defect categories are encoded using 1-of-6-vectors. In the recall phase, an actual output vector merely has to satisfy a much weaker condition in order to represent a particular defect category  $k$ :

$$(y_k \geq 0.9) \wedge (y_l \leq 0.1, l \neq k). \quad (9)$$

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:

- $h = 30$ ,
- $\alpha = 0.6, \beta = 0.1$ , and  $\gamma = \xi = 0.2$ ,
- $e_0 = 5$  and  $E_{max} = 0.5$ ,
- $\tau = 0.65$ .

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

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   |

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

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 back-propagation network [5] – an architecture often used in PD analysis (e.g. [12]). 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 symbolized by a 'firing'  $G_r$

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   |

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  |

neuron or by the fact that condition (9) is not satisfied.

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.

#### IV. CONCLUSIONS

An approach for neural network based classification of multi-dimensional real-valued data has been presented. The network – which has been named Guarded Neural Classifier – is an extension of a forward-only counterpropagation network. It has its name from two additional neurons, the ‘guards’, taking care of outliers in the training phase and 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 ‘unguarded’ 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. Developing a mechanism offering multiple output values *between* 0 and 1, estimating, for each output class, the probability the current input pattern belongs to that class – instead of merely yielding *one* additional binary value indicating whether or not the output is reliable – is an interesting project to be dealt with in the future.

#### REFERENCES

- [1] K. Liano, “A robust approach to supervised learning in neural networks,” in *Proc. of the IEEE Int’l Conf. on Neural Networks*, 1994, vol. 1, pp. 513–516.
- [2] C. Zhang, P. M. Wong, and O. Selinus, “A comparison of outlier detection methods: exemplified with an environmental geochemical dataset,” in *Proc. of the 6th Int’l Conf. on Neural Information Processing*, 1999, vol. 1, pp. 183–187.
- [3] J. Liu and P. Gader, “Outlier rejection with MLPs and variants of RBF networks,” in *Proc. of the 15th Int’l Conf. on Pattern Recognition*, 2000, vol. 2, pp. 680–683.
- [4] P. L. Rosin and F. Fierens, “Improving neural network generalisation,” in *Proc. of the Int’l Geoscience and Remote Sensing Symposium*, 1995, vol. 2, pp. 1255–1257.
- [5] 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*, 1986, vol. 1, pp. 318–362, MIT Press.
- [6] J.T. Lo and D. Bassu, “Training multilayer perceptrons in the presence of measurement outliers,” in *Proc. of the 2001 Int’l Joint Conf. on Neural Networks*, Vol. 3, 2001, pp. 2030–2035.
- [7] R. Hecht-Nielsen, “Counterpropagation networks,” *Applied Optics*, vol. 26, pp. 4979–4984, 1987.
- [8] B. Freisleben, M. Hoof, and R. Patsch, “Using counterpropagation neural networks for partial discharge diagnosis,” *Neural Computing & Applications*, pp. 318–333, July 1998.

- [9] T. Kohonen, *Self-organization and associative memory*, Springer–Verlag, 1989.
- [10] S. Grossberg, “Embedding fields: A theory of learning with physiological implications,” *Mathematical Psychology*, vol. 6, pp. 209–239, 1969.
- [11] M. Hoof, *Pulse-sequence-analysis: A new method of partial discharge diagnosis (in German)*, PhD Thesis, Universität-GH Siegen, Germany, 1997.
- [12] 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.