J29

# An Efficient Constrained Training Algorithm for Feedforward Networks

Dimitris A. Karras, *Associate Member, IEEE,* and Stavros J. Perantonis

*Abstract*—**A novel algorithm is presented which supplements the training phase in feedforward networks with various forms of information about desired learning properties. This information is represented by conditions which must be satisfied in addition to the demand for minimization of the usual mean square error cost function. The purpose of these conditions is to improve convergence, learning speed, and generalization properties through prompt activation of the hidden units, optimal alignment of successive weight vector offsets, elimination of excessive hidden nodes, and regulation of the magnitude of search steps in the weight space. The algorithm is applied to several small- and large-scale binary benchmark training tasks, to test its convergence ability and learning speed, as well as to a large-scale OCR problem, to test its generalization capability. Its performance in terms of percentage of local minima, learning speed, and generalization ability is evaluated and found superior to the performance of the backpropagation algorithm and variants thereof taking especially into account the statistical significance of the results.**

## I. Introduction

FEEDFORWARD neural networks (FNN's) have been the subject of intensive research efforts in recent years because of their interesting learning and generalization properties and their applicability in a variety of classification, approximation, and control problems. Following the backpropagation (BP) algorithm, a multitude of supervised learning algorithms have been devised in recent years with the aim of improving key properties of these networks, such as convergence ability, learning speed, scalability, and generalization capability. Each of these attempts has been focused on a single target, such as improvement of learning speed or generalization ability, but not on both of them. The key issue, however, is the development of training algorithms that simultaneously improve convergence ability, learning speed, and generalization capability in real-world large-scale problems.

A common objective of the BP algorithm and its descendants is to adapt the synaptic weights until the activations of the network's output layer units match prespecified values/targets. Apart from this *sine qua non* condition, some algorithms incorporate in their formulation additional information about FNN desired learning properties. For example, attempts to increase learning speed by helping the hidden units to play an active role during training, as well as attempts to improve generalization by enabling the decay of unnecessary synaptic weights, have been reported in the literature.

In this paper, the view is put forward that a significant amount of information about FNN desired learning properties can be incorporated during their training phase in the form of additional conditions on the weights and/or outputs of the network units. Following a critical overview of supervised learning algorithms, an algorithm for learning efficiently with constrained optimization techniques (ALECO) is presented which achieves a lower number of local minima, higher learning speed, and better generalization capability than other well-known supervised learning algorithms by incorporating information about the desired behavior of the hidden units and synaptic weights.

In the experimental part of this paper all important issues of FNN learning, such as learning speed, convergence ability, scalability, and generalization ability, are considered. In the effort of evaluating the performance of our algorithm and comparing it with that of other well-known supervised learning algorithms, we carefully discuss the selection of suitable algorithms for comparison, the choice of benchmarks, the organization of our experiments, and the presentation of our results. We thus propose a uniform empirical environment which enables fair comparison of the algorithms. Special attention is paid to ensuring statistical significance of the results for all algorithms and all benchmarks tried. On evaluating the performance of our algorithm, in terms of all the above mentioned critical issues of FNN learning, we find that it is an effective training procedure compared with BP and some of its well-known descendants.

This paper is organized as follows. In Section II, we present a critical overview of training algorithms proposed by other authors to improve key properties of FNN learning. Led by this discussion, we consider functional conditions that contain valuable information about FNN desired learning properties in Section III. These conditions codify the goals of avoiding local minima, accelerating learning, and improving generalization capability. Moreover, they lead to an optimization problem which forms the basis for ALECO. This optimization problem is formulated in Section IV, while ALECO is introduced and derived in Section V. In Section VI, we discuss the problem of evaluating and comparing the performance of different supervised learning algorithms for FNN. In the light of this discussion, we compare the performance of ALECO, in terms of convergence ability, learning speed, and generalization capability, with that of other popular and reputedly successful supervised learning algorithms in Section VII. Finally, Section VIII is an account of our conclusions.

Manuscript received May 5, 1993; revised September 24, 1994 and July 8, 1995.

The authors are with the Institute of Informatics and Telecommunications, National Research Center "Demokritos," 153 10 Aghia Paraskevi, Athens, Greece.

IEEE Log Number 9415041.

## II. Overview

A very important result of the studies related to FNN is the rigorous theoretical establishment, based on Kolmogorov's theorem [1] and Sprecher's modification [2], that these networks are universal approximators, once properly trained [3]–[5]. The problem of devising efficient algorithms for training FNN is thus of central importance to neural-network research. Among these algorithms, BP [6], [7] is the first one used and probably the most widespread, despite criticism that it is too slow and poor in terms of scalability (see, e.g., [8]) and generalization ability in certain difficult problems (see. e.g., [9]), too general to be used in a wide variety of problems [10], and biologically implausible [11]. This technique is suitable for application in the case of the complex FNN architecture, where the desired target output of every hidden layer unit is unknown. The algorithm has a well-defined objective (matching of the network outputs with the desired targets) which can be clearly formulated as a problem in mathematical terms (minimization of a cost function $E(\mathbf{T}, \mathbf{O}(\mathbf{w}))$, where $\mathbf{T}$, $\mathbf{O}$ are the desired and actual output vectors, respectively, the latter considered as functions of the weight vector $\mathbf{w}$). The problem is solved via search in the weight or the transformed weight space [10] using a mathematically rigorous and well-defined optimization technique (gradient descent), whose convergence properties have been thoroughly studied [12], [13]. Note, however, that BP does not take into account any additional information about learning in a FNN, apart from the multilayer architecture and the desired input-target output relation.

Regarding generalization ability, especially in classification problems with noisy and low-dimensional data [14], the BP algorithm often exhibits very good performance, better than that of several conventional classifiers [15]. The major drawbacks of this algorithm, however, are its slow learning speed and occasional convergence to local minima [16]. Occurrence of local minima hampers training and results in attaining inferior classification [17] or function approximation performance. In addition, the two problems of slow learning speed and convergence to a certain kind of local minima are closely related and can be partially attributed to the issue of premature saturation [16]. Therefore, the problems of generalization ability, convergence to local minima, and slow learning speed are related to one another. Thus, due attention should be paid to these three major issues simultaneously if we aim at designing reliable FNN classifiers/controllers/predictors. Concerning especially the problem of slow learning speed, one might tend to consider it a minor issue compared to generalization capability. Some recent solid theoretical and experimental results, however, have shown that the problem of loading a set of training examples onto a neural network is NP-complete [18]–[20]. Thus it is unlikely that existing algorithms (like BP) can be guaranteed to learn the optimal weight solution in polynomial time. This problem is caused by several specific characteristics of the FNN error surfaces as discussed in Section III. These results show clearly the severity of the slow learning speed issue in FNN and are largely confirmed in recent work in satellite imaging classification tasks [21], without degrading generalization performance, and ability of convergence to global minima still remain very important research goals and due attention is paid to them in this paper.

FNN supervised learning algorithms which have been proposed after BP with the aim of improving learning speed and generalization ability can be classified into two main categories with respect to their approach of the learning problem. The first category comprises all algorithms which consider supervised learning in a given FNN simply as the unconstrained minimization of one cost function $E(\mathbf{T}, \mathbf{O}(\mathbf{w}))$ in the weight or transformed weight space. This function is encountered in many forms such as the conventional sum of squared errors, the hyperbolic arctangent [8] form, and the "entropic" like form [23], etc. This first category includes all algorithms originating from the field of numerical analysis such as second-order methods [24], [25], line search methods (e.g., steepest descent and conjugate gradients [26]), and quasi-Newton methods [27]. It also includes all algorithms originating from the field of optimal filtering (e.g., extended Kalman algorithm [28]). Finally, all heuristic optimization techniques which perform a search in the weight space (e.g., Quickprop [8] and Delta-Bar-Delta [29]) belong to this category.

Criticism about some of these techniques can be found in [8], [30], and [31], but we wish to stress an entirely different point. In our view, the common characteristic of these approaches is that they identify the learning problem with the unconditional optimization procedure of one function. In this procedure, information about the architecture of the system and the desired input–output association is incorporated but not much else. It is thus assumed that optimization methods proved to be successful in the fields of numerical analysis or optimal filtering will preserve their efficiency if they are slightly modified, or even worse, copied [31] and then applied to the problem of learning. The efficiency of a method, however, depends on its inherent capability to incorporate into its formalism as many properties of the field to which it is applied as possible. From this point of view, all the above procedures suffer from a very serious drawback. They do not include in an integrated manner the continuously accumulated information about FNN learning, some aspects of which we address in the following paragraphs.

The second category of supervised algorithms for solving the FNN learning problem includes algorithms embodying in their formalisms additional information about learning in FNN. Along these lines, different strategies have been presented. For example, le Cun criticizes in [10] the notion of a general BP architecture that can solve the learning problem in every case. The concept of task-specific architectures is therefore introduced. Although this way of building information about desired learning properties into an FNN system has been proved successful in the case of image recognition [32], it is very difficult to specify the appropriate architecture in the general case [10].

Another strategy for handling the information incorporating problem includes many algorithms based on a mixed cost function approach, where optimization of many criteria

some algorithms are based on the key role of hidden units in the achievement of the desired input–output mapping during training. The learning process has been described by many authors (see, e.g., [6], [33], and [34]) in terms of finding good "internal representations" of the input patterns on the hidden units and several algorithms have been proposed attempting prompt activation of these units. In a group of related algorithms, supervised learning in FNN is defined as minimization of $E(\mathbf{T}, \mathbf{O}(\mathbf{w}, \mathbf{S}))$ where $\mathbf{S}$ is the vector of the internal representations. Information about the behavior of the hidden units is thus incorporated in the cost function. Minimization procedures used can be formal, as in the moving targets algorithm [35], or heuristic, as in the CHIR algorithm [36], [37]. In a final class of such algorithms, the information about desired learning properties is explicitly presented in the form of penalties added to the cost function. Learning is described as minimization of $E(\mathbf{T}, \mathbf{O}(\mathbf{w}))+kF(\mathbf{w}, \mathbf{S})$ [38] or as minimization of $E(\mathbf{T}, \mathbf{S}, \mathbf{w})+kF(\mathbf{w}, \mathbf{S})$ [30], where $k \in \mathcal{R}$ and $F(\mathbf{w}, \mathbf{S})$ is a suitably defined function of the weights and hidden node outputs.

It should be mentioned that most of these methods lack the clarity of BP in both objectives and methodology. The search space is extended beyond the initial weight space to include the hidden unit outputs without due attention being paid to the relations among these variables imposed by the FNN architecture; often heuristic approaches are adopted to reach acceptable solutions. Moreover, no study has been presented which shows an advantage of these methods over other well-established training algorithms—including BP—concerning learning speed or generalization capabilities. To be more specific, it turns out [30] that the only slight improvement resulting from these algorithms, compared with the classical BP, lies in the case of local minima. Since most of these algorithms extend the search space by increasing the number of independent free parameters of the learning problem, we may expect that generalization results will be relatively poor [10], [39] and learning times will increase. Only some more carefully formulated algorithms, which are based on the mixed cost function approach but do not extend the search space, exhibit success as regards improvement of generalization performance. Examples are weight elimination techniques [40], [30], where learning is described as minimization of $E(\mathbf{T}, \mathbf{O}(\mathbf{w})) + kF(\mathbf{w})$ and the double BP algorithm [41], where learning is described as minimization of $E(\mathbf{T}, \mathbf{O}(\mathbf{w}))+kF(E(\mathbf{T}, \mathbf{O}(\mathbf{w})), \mathbf{I})$ ( $\mathbf{I}$ is the vector of external inputs).

Although the mixed cost function method is frequently encountered in the optimization literature [42], its major drawback is that it partly compromises the main objective of FNN training phase which is to adapt the synaptic weights until the activations of the network's output layer units match prespecified values/targets. Thus, this mixed BP-output-error function formalism does not mathematically guarantee reduction of the traditional BP output error function until its local or global minima are reached, as the conventional BP formalism does. Another drawback of this mixed BP-output-error function optimization method is the large number of parameters which should be tuned during learning, especially when the goal is to embody a lot of information about desired learning properties in FNN training, in the form of many functional conditions.

The overview presented in this section raises two important questions, which we take up next. Our answers to these questions lead directly to the formulation of ALECO:

- Which types of additional information about learning in FNN would it be of interest to incorporate into training algorithms in the form of suitable functional conditions? This issue is taken up in detail in Section III.
- What methods are appropriate for incorporating such information into FNN training algorithms? Methods are needed which maintain the clarity in objectives, rigor, and mathematical elegance of BP and overcome the drawbacks of the mixed cost function optimization approach. Such methods can be sought in the rigorous frameworks of nonlinear programming and optimal control. In particular, ALECO will be based on a modification of the gradient ascent technique proposed by Bryson and Denham in the early 1960's [43], as discussed in Sections IV and V.

## III. Functional Conditions for Improving FNN Learning Properties

In this section, we formulate functional conditions under which training in FNN is demanded to take place along with the minimization of the well-known mean square error cost function. These conditions incorporate information about learning, in the form of goals for avoiding local minima, increasing learning speed, and improving generalization capability, and are integral parts of the ALECO training procedure which will be derived in Section V.

Consider an FNN with one layer of input, $M$ layers of hidden, and one layer of output units. The units in each layer receive input from all units in the previous layer. We denote the unit outputs by $O_{ip}^{(m)}$, where the superscript $(m)$ labels a layer within the structure of the neural network ($m = 0$ for the input layer, $m = 1, 2, \cdots, M$ for the hidden layers, $m = M+1$ for the output layer), $i$ labels a unit within a layer, and $p = 1, 2, \cdots, R$ labels each of $R$ input patterns. The synaptic weights are denoted by $w_{ij}^{(m)}$, where $m$, $j$ denote, respectively, the layer and the unit toward which the synapse is directed and $i$ denotes the unit in the previous layer from which the synapse emanates. Biases will be treated as weights emanating from units with constant, pattern-independent output equal to one.

### A. Alignment of the Hidden Unit Outputs with the Output Layer Errors

This desirable property has been introduced by Fahlman in his cascade correlation algorithm [44]. Here we use as a measure of this alignment the following function

$$\Phi_1 = \frac{1}{2R}\left[\sum_{ijpm}\left(\varepsilon_{jp} - \langle\varepsilon_{jp}\rangle_p\right)^2\left(O_{ip}^{(m)} - \langle O_{ip}^{(m)}\rangle_p\right)^2\right]^{1/2} - E/R \tag{1}$$

which bears some similarity to that used by Fahlman. This function should be increased as training progresses. The symbol $\langle\,\rangle_p$ denotes averaging with respect to $p$ and the quantities $\varepsilon_{jp}$ denote the squared output layer errors. We argue that incorporation in the training algorithm of a condition which leads to such an alignment can have beneficiary effects upon reduction of local minima incidence rate. In an analysis of the frequently encountered types of local minima in FNN training [45], the physical correlates of the most prominent classes of these local minima have been described as follows:

- Type 1: Hidden nodes whose decision boundaries have been moved outside the training sample region in the sample space (stray hidden nodes).
- Type 2: Hidden nodes that implement the same function.
- Type 3: Hidden nodes that are all inactive in a specific region of the training sample space.

These physical correlates of the local minima can be related to several factors [17]:

- Local minima incidence, especially at the beginning of learning, can be sometimes attributed to the existence of hidden units that have gradual transition regions. These units are sensitive to the training process and therefore are desirable. In these regions, the hidden units tend to approximate linear transfer functions. The combined outcome of a pair of such transfer functions is only another different linear function, which could have been produced by a single hidden node. Therefore, the decision boundaries of each such hidden node are obscured by this combined functioning. It often happens that either both hidden units duplicate one function or the one implements the necessary function, while the other tends to move out of the sample region [17]. As a result, local minima of type 1 or type 2 are created. By aligning hidden node outputs to the output errors at each epoch in the initial stages of training when the output error is large, we help input-to-hidden layer weights to evolve rapidly and drive hidden nodes to activate at the nonlinear regions of the transfer function, thus avoiding local minima originating from the undesirable effects of linearity.
- Local minima incidence is also related to amplification of error signals propagating backward from the output units due to the existence of large hidden-to-output weights. The input-to-hidden layer weights are thus dramatically perturbed [17], since their adaptation rule is closely related to the magnitude of the hidden-to-output node weights. This results in large shifts in the positions of the decision boundaries of these hidden units and therefore creates local minima of types 1 or 3. Alignment of as many hidden node outputs as possible to the output errors at each epoch can hinder the oscillation and extensive shifting of the hidden nodes: When the output error is gracefully reduced, hidden node activations are also gracefully stabilized near their mean value and therefore, the input-to-hidden layer weights are forced to gracefully adapt. Thus the disastrous effects in decision boundaries of hidden units that lead to certain local minima can be avoided.

### B. Alignment of Weight Updates in Successive Epochs

FNN error surfaces have certain characteristics that make them very difficult to search. In specific, they have many long narrow troughs flat in one direction and steep in surrounding directions [46]. In the vicinity of such regions, gradient descent proper suffers from very high frequency variations in weight space, settles into zig-zag trajectories, and is hopelessly slow [42], [47]. Supplementing gradient descent with momentum acceleration represents a compromise between the need to decrease the cost function at each epoch and the need to proceed along relatively smooth paths in the weight space. The formalism favors configurations where the current and previous weight update vectors are partially aligned, thus avoiding zig-zag paths and accelerating learning. Evidence of such behavior in small networks is given in [48]. The use of a constant momentum coefficient, representing a constant, *a priori* chosen degree of weight update alignment, however, does not help avoid zig-zag trajectories in troughs of arbitrary eccentricities [47]. A scheme for the momentum term, which results in adaptive rather than *a priori* alignment between current and previous weight vector changes, according to the specific characteristics of error surface localities, has been successfully investigated in [48] and found to achieve increased learning speed. In this work, the following expression was used as a measure of the alignment of successive weight offsets, which should be increased at each epoch during learning

$$\Phi_2 = \sum_{ijm} \left( w_{ij}^{(m)} - w_{ij}^{(m)}\Big|_c \right) \left( w_{ij}^{(m)}\Big|_c - w_{ij}^{(m)}\Big|_l \right). \quad (2)$$

In the above equation $l$ and $c$ mean last and current epoch, respectively.

From the above discussion we are led to the conclusion that imposition of adaptive alignment between current and previous weight vector changes on FNN training should lead to increased learning rates. This expectation is confirmed in [48] and in the experimental study of this paper.

### C. Elimination of Excessive Hidden Nodes

Decrease of the function

$$\Phi_3 = \sum_{im} \left( \langle O_{ip}^{(m)} \rangle_p \right)^2, \quad 1 \le m \le M \quad (3)$$

at each epoch aims at making the average outputs of as many hidden units as possible very small during convergence of the training procedure, so that they do not play an active role in the feedforward propagation of the input signals. In this way, some hidden units are eliminated, with an expected improvement in generalization capability due to the reduction of the effective number of model parameters and thus reduction of the generalization error because of the bias of the model [49]. By using this pruning procedure we can automatically determine the maximum number of necessary hidden units for the successful termination of the FNN training process, thus reducing the need for extensive experiments to find the best hidden layer architecture.

*D. Measure of the Weight Update Vectors Equal to a Constant*

We impose on FNN learning the condition that the sum of squares of the individual weight changes take a predetermined positive value $(\delta P)^2$

$$\Phi_4 = \sum_{ijm} \left( w_{ij}^{(m)} - w_{ij}^{(m)}\Big|_c \right)^2 - (\delta P)^2 = 0. \tag{4}$$

Thus, at each epoch, the search for an optimum new point in the weight space is restricted to a small hypersphere of radius $\delta P$ centered at the point defined by the current weight vector. The main objective of this condition is improvement of the FNN generalization capability. In the BP algorithm, the cost function landscape is complicated with many flat regions and long narrow troughs [47] and can be difficult to search efficiently. Global minima regions can be located in such troughs, and thus this searching inefficiency cannot lead to good generalization performance. We can understand these BP searching deficiencies easily, if we think of how this algorithm attempts to overcome very slow learning rates in flat regions. Because cost function gradients are very small in flat regions, it is imperative to use large learning rates if we want to complete the training process in reasonable time. When steep regions with large cost function gradients are encountered, however, this strategy leads to abrupt long jumps so that interesting regions can be missed altogether. To overcome these difficulties, we propose condition (4) which aims at eliminating the long abrupt jumps in weight space and thus searching the vicinity near global optima more efficiently. Therefore, we should expect better generalization capabilities in FNN by imposing this condition.

## IV. Formulation of the Optimization Problem

We view supervised learning as an iterative process with the following objectives:

- Reach, upon completion of the iterative process, a minimum of the cost function

$$E = \frac{1}{2} \sum_{ip} \left( O_{ip}^{(M+1)} - T_{ip} \right)^2 \tag{5}$$

with respect to the synaptic weights and output activations ($T_{ip}$ represent target values for units in the output layer of the FNN). To this end, the cost function is decremented at each iteration (epoch) by an amount $\delta Q$. This should be chosen adaptively, so that the accumulated changes to the nonnegative cost function after a number of epochs should suffice to achieve the desired input–output relation.

- In addition, certain conditions should be satisfied at each epoch of the training algorithm. These include the architectural constraints, as well as additional conditions representing desirable information about the learning process. In particular, the following should be met:

1) The architectural constraints

$$f_{ip}^{(m)}(\mathbf{O}, \mathbf{w}) = g\left( \sum_i w_{ij}^{(m)} O_{ip}^{(m-1)} \right) - O_{jp}^{(m)} = 0 \tag{6}$$

should be satisfied. Here $g$ is the logistic function $g(x) = 1/(1 + \exp(-x))$ and biases are treated as weights emanating from nodes of constant, pattern-independent activation equal to one.

2) Maximum change should be achieved for $\Phi = \Phi_1 + \Phi_2 + \Phi_3$, thus favoring correlation of the hidden unit outputs to output layer errors, enhancement of the alignment of successive weight updates, and elimination of excessive hidden nodes.

3) Finally, the condition $\Phi_4 = 0$ should be satisfied.

If $\delta P$ is small enough, the changes to $E$ and $\Phi$ induced by changes in the weights can be approximated by the first differentials $dE$ and $d\Phi$. This problem is related to the gradient ascent method for solving nonlinear programming problems proposed by Bryson and Denham [43]. Recent attempts along the same direction can be found in [50] and [51]. We note that the weight changes at each epoch of the learning process, considered as the independent variables, can be determined by solving analytically the following constrained optimization problem.

Maximize, with respect to all $dw_{ij}^{(m)}$ the function $d\Phi$, under the constraints

$$dE = \delta Q \tag{7}$$

$$\Phi_4 = 0 \tag{8}$$

and

$$f_{ip}^{(m)} = 0. \tag{9}$$

## V. Derivation of ALECO

The constrained maximization problem defined by (7)–(9) can be solved by considering the admissible variations of $\Phi$ and $E$, $d\Phi$ and $dE$, respectively, subject to the architectural constraints (9). This can be achieved by introducing suitable Lagrange multipliers. Hence, we construct the functions

$$e = E + \sum_{jpm} \lambda_E^{jp(m)} f_{jp}^{(m)} \tag{10}$$

$$\phi = \Phi + \sum_{jpm} \lambda_\Phi^{jp(m)} f_{jp}^{(m)} \tag{11}$$

where the $\lambda_E$ and $\lambda_\Phi$ are Lagrange multipliers to be determined in due course.

Consider the differentials

$$de = \sum_{jpm} \frac{\partial e}{\partial O_{jp}^{(m)}}\bigg|_c dO_{jp}^{(m)} + \sum_{ijm} \frac{\partial e}{\partial w_{ij}^{(m)}}\bigg|_c dw_{ij}^{(m)} \tag{12}$$

$$d\phi = \sum_{jpm} \frac{\partial \phi}{\partial O_{jp}^{(m)}}\bigg|_c dO_{jp}^{(m)} + \sum_{ijm} \frac{\partial \phi}{\partial w_{ij}^{(m)}}\bigg|_c dw_{ij}^{(m)}. \tag{13}$$

We choose the $\lambda_E$ and $\lambda_\Phi$ so as to eliminate all dependence of $de$ and $d\phi$ on the $O_{jp}^{(m)}$

$$\frac{\partial e}{\partial O_{jp}^{(m)}}\bigg|_c = 0, \quad \frac{\partial \phi}{\partial O_{jp}^{(m)}}\bigg|_c = 0. \tag{14}$$

This leads to closed formulas for determining the Lagrange multipliers. From (6), (10), and (11) we readily obtain

$$\lambda_E^{jp(M+1)} = O_{jp}^{(M+1)}\Big|_c - T_{jp} \tag{15}$$

$$\lambda_\Phi^{jp(M+1)} = \frac{\partial \Phi}{\partial O_{jp}^{(M+1)}}\Big|_c \tag{16}$$

$$\lambda_E^{ip(m)} = \sum_j \lambda_E^{jp(m+1)} w_{ij}^{(m+1)}\Big|_c \; O_{jp}^{(m+1)}\Big|_c \left(1 - O_{jp}^{(m+1)}\Big|_c\right). \quad m = 1, 2, \cdots, M \tag{17}$$

$$\lambda_\Phi^{ip(m)} = \frac{\partial \Phi}{\partial O_{ip}^{(m)}}\Big|_c + \sum_j \lambda_\Phi^{jp(m+1)} w_{ij}^{(m+1)}\Big|_c \; O_{jp}^{(m+1)}\Big|_c \left(1 - O_{jp}^{(m+1)}\Big|_c\right), \quad m = 1, 2, \cdots, M. \tag{18}$$

The Lagrange multipliers can thus be determined in the following systematic way: Multipliers corresponding to the output layer are evaluated. Multipliers of the $m$th layer are readily determined once the ones corresponding to the $(m+1)$th layer have been evaluated. This procedure can be considered as a "backpropagation" of the Lagrange multiplier values.

Differentiating (10) and (11) with respect to the synaptic weights and having eliminated all dependence on $O_{ip}^{(m)}$, we obtain the following equations for points satisfying the architectural constraints

$$dE = de = \sum_{ijm} J_{ijm} dw_{ij}^{(m)},$$
$$d\Phi = d\phi = \sum_{ijm} F_{ijm} dw_{ij}^{(m)} \tag{19}$$

with

$$J_{ijm} = \sum_p \lambda_E^{jp(m)} O_{jp}^{(m)}\Big|_c \left(1 - O_{jp}^{(m)}\Big|_c\right) O_{ip}^{(m-1)}\Big|_c \tag{20}$$

$$F_{ijm} = w_{ij}^{(m)}\Big|_c - w_{ij}^{(m)}\Big|_l + \sum_p \lambda_\Phi^{jp(m)} O_{jp}^{(m)}\Big|_c \left(1 - O_{jp}^{(m)}\Big|_c\right) O_{jp}^{(m-1)}\Big|_c. \tag{21}$$

We are ready now to return back and analytically solve the constrained optimization problem of (7)–(9) at each epoch, for the points satisfying the architectural constraints. To this end, we introduce Lagrange multipliers $\lambda_1$ and $\lambda_2$ to take account of the remaining constraints in this problem [(7) and (8)]

$$d\phi = d\Phi = \sum_{ijm} F_{ijm} dw_{ij}^{(m)} + \lambda_1 \left(\delta Q - \sum_{ijm} J_{ijm} dw_{ij}^{(m)}\right) + \lambda_2 \left[(\delta P)^2 - \sum_{ijm} dw_{ij}^{(m)} dw_{ij}^{(m)}\right]. \tag{22}$$

Note that the quantities multiplying $\lambda_1$ and $\lambda_2$ are equal to zero by (7), (8), and (19) and that $\delta P$ and $\delta Q$ are known quantities. We obtain maximum change in $\Phi$ at each iteration of the algorithm by ensuring that

$$d^2\Phi = \sum_{ijm} \left(F_{ijm} - \lambda_1 J_{ijm} - 2\lambda_2 dw_{ij}^{(m)}\right) d^2 w_{ij}^{(m)} = 0 \tag{23}$$

$$d^3\Phi = -2\lambda_2 \sum_{ijm} d^2 w_{ij}^{(m)} \cdot d^2 w_{ij}^{(m)} < 0. \tag{24}$$

To satisfy (23) we set

$$dw_{ij}^{(m)} = -\frac{\lambda_1}{2\lambda_2} J_{ijm} + \frac{1}{2\lambda_2} F_{ijm}. \tag{25}$$

The coefficients of $J_{ijm}$ and $F_{ijm}$ are changed adaptively. To see this, we use (8), (19), and (25) to obtain

$$\lambda_2 = \frac{1}{2}\left[\frac{I_{EE}(\delta P)^2 - (\delta Q)^2}{I_{\Phi\Phi} I_{EE} - I_{E\Phi}^2}\right]^{-1/2}$$
$$\lambda_1 = (I_{E\Phi} - 2\lambda_2 \delta Q)/I_{EE} \tag{26}$$

where

$$I_{EE} = \sum_{ijm} (J_{ijm})^2 \tag{27}$$

$$I_{E\Phi} = \sum_{ijm} J_{ijm} F_{ijm} \tag{28}$$

$$I_{\Phi\Phi} = \sum_{ijm} (F_{ijm})^2 \tag{29}$$

and the positive square root value was chosen for $\lambda_2$ to ensure maximum (rather than minimum) $d\Phi$ (relation 24).

Note the bound $|\delta Q| \le \delta P \sqrt{I_{EE}}$ set on the value of $\delta Q$ by (26), which forces us to choose $\delta Q$ adaptively. The simplest choice for adapting $\delta Q$, namely

$$\delta Q = -\xi \delta P \sqrt{I_{EE}}, \qquad 0 < \xi < 1 \tag{30}$$

is most attractive because of its learning convergence properties. Indeed, as in BP, it is possible to show for small enough $\delta P$ that the algorithm converges to global or local minima of the cost function of (5). To see this, we use (12), (19), and (27) to rewrite (30) as follows

$$\delta Q = -\xi \delta P \left[\sum_{ijm} \left(\frac{\partial e}{\partial w_{ij}^{(m)}}\Big|_c\right)^2\right]^{1/2}. \tag{31}$$

Hence, it suffices to show that for a given positive real number $\eta$ there exists an epoch number $\nu_0$, so that $|\partial e/\partial w_{ij}^{(m)}| < \eta$ for all subsequent epochs. Indeed, in the opposite case, an infinity of changes in $E$ at least equal to $-\xi\eta\delta P$ would accumulate and drive $E$ to minus infinity as learning progressed. This is not possible, since $E$ is bounded from below by zero.

In short, (25) describes the weight updating formula of ALECO, which optimizes the weight steps in each epoch and converges to minima of the cost function. Taking into account (30) we are left with two free learning parameters $\delta P$ and $\xi$ which should be adjusted to achieve optimum performance.

## VI. Experimental Study: Methodology

At present, there is no general agreement in the neural network community on how to measure the performance of the various training algorithms in terms of either their speed [8], [52] or their generalization capability. In this section we address five major aspects of this methodology problem: Selection of algorithms for comparison, benchmark selection, organization of the experiments, means of attaining adequate statistical significance for the results, and selection of learning parameters. Our experimental study is carefully designed and carried out in the light of this discussion on methodology.

### A. Selection and Implementation of Algorithms

Although it is impossible to compare ALECO with the vast number of supervised learning algorithms proposed in the literature, we have chosen to compare it not only with BP (off-line and on-line version), but also with algorithms which reputedly improve its performance significantly. We have selected representative, well-known, and reputedly successful algorithms which have fixed and predefined model architecture—so we have excluded methods like cascade correlation [44], whereby the network size is expressly varied during learning. These algorithms are the Quickprop [8] and Delta-Bar-Delta [29] learning rules.

We have used our own FNN algorithm simulation program developed at NCSR "Demokritos," which is described in detail in [53]. Algorithms were implemented on the SUN-sparcstation network of NCSR "Demokritos" for small-scale benchmarks, while for large-scale benchmarks we used the Silicon Graphics Crimson as well as the supercomputer Convex 3820 of the same research center.

### B. Benchmark Selection and Description

At present, the research effort for devising appropriate benchmark models to compare different training algorithms is at its beginning, and there is no wide consensus about the proposed benchmarks. Nevertheless, the general tendency in the literature is employment of artificial binary benchmarks (see, e.g., [8]) and large-scale real-world classification tasks (see, e.g, [32]). Binary benchmark problems—encoder [8], [23], [52], [54], counter [23], and multiplexer [29]—are the ones that most authors are in favor of, especially in the case of training algorithm comparisons with respect to their convergence abilities and learning speed. An attractive feature of these benchmarks is their flexibility: Simple algorithms can be used to produce encoders, counters, and multiplexers of arbitrary size, complexity, and difficulty, thus providing strict tests for convergence abilities, learning speed, and scalability properties. Another characteristic of these tasks is that, although artificial, they give a good insight about the performance of training algorithms in real-world tasks. This is especially true for encoders [8]. Real-world classification problems are considered mainly when the primary interest is in generalization capability of the training algorithms [32], [33], [55]. Among them, OCR-related problems are the most widely used. In addition, it is very difficult to reach perfect generalization in OCR-related tasks and even a small improvement is of great practical importance—particularly when classification accuracy already achieved is very high—provided that statistical significance of the results is guaranteed.

In this experimental study we evaluate all training algorithms in three categories of experiments. The first category includes small-scale experiments, while the second includes large-scale problems involving either a large number of synaptic weights or a large number of patterns in the training set. In both categories we employ binary benchmarks and evaluate the performance of the algorithms concerning learning speed and avoidance of local minima. Finally, the third category includes a large-scale classification problem related to OCR, the objective being to evaluate generalization capabilities as well as convergence abilities of the training algorithms tried. In our small-scale experiments we include a counter (to be solved by a 4-5-5 FNN architecture), a multiplexer (6-6-1 architecture with 64 input patterns), and an encoder problem ("strict" encoder with 8-3-8 architecture). We also use XOR, the well-known and popular benchmark, because of historical reasons [56]. In general, these are the benchmarks used by the authors of the algorithms to which we compare ALECO. In this way we ensure performance evaluation of ALECO in the environment proved to be best for the other algorithms. On the other hand, in our large-scale experiments we include a multiplexer (11-11-1 architecture with 2048 input patterns) and two encoder problems (two "strict" encoders with 128-7-128 and 256-8-256 architectures, respectively).

Finally, the third category of our experiments involves an OCR-related task for lowercase Greek typeset characters. The main difficulties that a classifier has to overcome in OCR for typeset characters are random noise of unknown probability distribution and deformations due to deficiencies encountered in printing, photocopying, and scanning of documents. Problems due to rotation are not normally met. Therefore, this experimental study aims mainly at comparing the generalization capabilities of different FNN classifiers in the case of noisy (with no model assumption for noise) and deformed patterns.

In all OCR-related experiments we use the same preprocessing, segmentation, and normalization regulations. A filter is used to remove pixel regions under a certain threshold (noise regions), horizontal and vertical vacant lines segmentation is performed, and a $25 \times 25$ pixel normalization is used with the character width as the width boundary and the boundary of the text line as the height boundary.

Regarding feature extraction, the demand for preliminary filtering and efficient image encoding led directly to the employment of the Laplacian pyramid theory [57]. In the proposed OCR benchmark a one-level Gaussian pyramid was used in the feature extraction stage with the following characteristics:

- Two generating kernels (Gaussian masks), were used ($h_5$, a five-by-five mask, and $h_4$, a four-by-four mask). The pattern of weights $h(m, n)$ of the generating kernels was chosen subject to the constraints of normalization and equal contribution for the different levels [58], as well as equality for all weights at a given level.

- We have convolved each original rastered character image with the two above mentioned generating kernels, $h_5$ and $h_4$, separately. The input of the classifier is obtained by keeping the most important terms of the two convolution series. This scheme takes into account different correlations of the neighboring image pixels and achieves higher dimensionality reduction rates than the convolution scheme proposed in [57].

The feature extraction stage results in the construction of 41 convolutional features per character. These features are input to a fully connected FNN with two layers of weights and a 41-40-32 architecture. Using different training algorithms, the network is trained to assign the convolutional features to one of 32 character categories. The desired output of all output nodes is zero, except for the node associated with the category of the input character, whose desired output is one.

For reasons of fair comparison, all experiments were conducted using the same character sets for training and recognition. At each training trial, the final weights used for testing were those for which recognition accuracy in a "validation set" of characters had reached its maximum value during training. The training, validation, and test character set of continuous valued convolutional features were obtained as follows:

- The union of three artificial databases containing characters of arial, arc, and times fonts (total of 960 characters) forms our training set. Each of the arc, arial, and times MS Windows 3.1 database has been obtained by using an HP scanner at 300 dpi resolution, with the default contrast regulation in the scanning software, to binarize a text printed from an HP printer with 320 characters (32 classes with 10 prototypes each).
- Our validation set similarly comprised the union of arial, arc, and times artificial character databases (total of 960 characters) as in the training set above. Instead of using the default contrast regulation in the scanning software, however, we have utilized a different one for producing the validation character database by employing the same procedure as in the construction of the training set.
- Three natural sets of arc, arial, and times fonts (total of 4647 characters) and an artificial character set of courier fonts (total of 320 characters) were used as test sets. These recognition sets have the following specifications:

  **Text1 set:** The scanner, printer, and scanning software contrast regulation were different from those used in the training and validation sets. We had 1554 characters in a plain text of MS Windows 3.1 arial and arc fonts.

  **Text2 set:** We had different scanner, printer, and contrast regulation from those used in the training and validation sets. The character set consisted of 1508 characters in a plain text of MS Windows 3.1 times font.

  **Text3 set:** The scanner and printer were different from those used in the training and validation sets, but the contrast regulation was the same as in the training phase. We had 1585 characters in a plain text of MS Windows 3.1 arial font.

  **Courier set:** The scanner, printer, and scanning software contrast regulation were those used in the training phase. We had one database with 320 characters (32 classes with 10 prototypes each) of MS Windows 3.1 courier font. None of the FNN used in the test phase experiments has been trained with this character font. The aim of this experiment is to evaluate the generalization performance of the algorithms in comparison, in a completely different font than the ones included in the training set.

### C. Organization of the Experiments

Three major problems plague several authors' reports on experimental results in FNN learning. Often, insufficient information is provided for other researchers wishing to reproduce the experiments or compare the proposed method with others [52]. On other occasions, conclusions about algorithm performance are drawn from insufficient statistics. Finally, the comparison of reports is often difficult because of nonuniformity in convergence or generalization criteria. In effect, results quoted by different authors on the performance of the same training algorithm tested on the same training task sometimes differ by orders of magnitude (see, for example, the XOR comparison tests in the reports of [8] and [29] on the one hand and of [59] on the other). The crucial issue here is the task of building a uniform environment to fairly compare all algorithms. We believe that we would have more order in the chaos of neural-network comparison reports if this task had been considered by all authors as seriously as by Jacobs [29] and Fahlman [8].

Using the work of these authors as our starting point, we build a uniform environment for experimenting with FNN. Thus, all information relevant to a particular experiment is reported, which would be of use to other researchers wishing to reproduce our results or compare our algorithm to other methods. Benchmark or algorithm specific factors and parameter values are reported in the section on experimental results (Section VII) individually for each algorithm and benchmark. Moreover, certain factors and parameter values are kept the same for all algorithms and benchmarks to ensure uniformity and fair comparison. These are considered and reported as follows:

- The type of unit activation function used and its parameters. The importance of this factor can be found in many studies. For example, see [60] and [61]. We have used the sigmoid logistic function $g(x) = 1/(1+\exp(-x))$ as activation function for every FNN processing unit.
- The weight initialization procedure. The importance of this factor is described in [8]. We use random initial weights with uniform distribution in $[-r, r]$. A fixed $r, r = 0.5$, was used for all algorithms and for all experiments tried. At each individual trial, the same initial weights were used for all algorithms.
- The cost function to be minimized. The significance of the choice of this function is well documented in the literature [8] and [23]. For reasons of fair comparison, we have used the quadratic cost function (5) in all experiments performed.
- The order according to which the training set is presented at each epoch to the input layer of the network. The

significance of this parameter is discussed in [62] and [63]. In all experiments, we use a fixed order of pattern presentation for all epochs. For the binary type benchmarks, all patterns are presented sequentially indexed in their categories at each epoch.

- The convergence criterion. We have adopted Fahlman's 0.4–0.6 criterion as the most sensible for all benchmarks tried here [8].
- Other critical parameters used in the unit or net architecture. We have accepted Fahlman's analysis about eliminating the "flat spot" [8], and added the parameter $S' = 0.1$, whenever the product $O_{jp}^{(m)}(1 - O_{jp}^{(m)})$ (the derivative of the logistic function) was encountered for every experiment and algorithm. This had an accelerating effect on all algorithms.

### D. Statistical Significance of Experimental Results

Two factors are important for ensuring statistical significance of the experimental results, namely number of experimental trials, that is, number of training trials starting from different initial points in the weight space, and maximum allowed percentage of failures. The importance of the first factor is partially described in [8] and [52], while reporting of the second factor is first introduced in this work. Their role is analyzed next.

In the majority of FNN training algorithms, trials are started by random initialization of the weights. The search for a minimum of the cost function varies in terms of difficulty at each trial. Therefore, it is very important to test our algorithms for a sufficient number of trials to ensure an adequate level of statistical significance for our results and a fair comparison of different training algorithms.

The ratio $P_s$ of successful to total number of learning trials is closely related to the problem of local minima. Since all presently known successful training algorithms are based essentially on local optimization techniques of the almost never convex cost function, there is no warranty of convergence at the global optimum for every training trial. Therefore, one should decide *a priori* about an acceptable level of allowed failures of convergence especially if learning speed and avoidance of local minima is of primary concern. The learning parameters should then be chosen to achieve both this level and the best possible results in the successful trials. In our experiments we have always chosen suitable parameters to achieve a minimum of $P_s = 0.70$.

We propose that a test of statistical significance should be performed at least on the average number of epochs needed to successfully complete a training task, as well as on the average classification accuracy obtained by each algorithm.

Concerning the small- and large-scale experiments used to test learning speed, we have been able to perform enough experimental trials to ensure at least with 99% probability that the experimental average number $\mathcal{M}$ of epochs needed to successfully complete each task differed from the true average by less than 10%. At this level of statistical significance, we can obtain a good quantitative measure for the relative learning speed improvement offered by ALECO over other training algorithms. To determine the minimum number $\nu_{\min}$ of training trials required to achieve this level of statistical significance, we have used the central limit theorem of probability theory. Indeed, it can be shown (see the Appendix) that an estimate of $\nu_{\min}$ required to obtain a value of $\mathcal{M}$ differing from the true expected value $\bar{\mathcal{M}}$ of the distribution of the number of epochs by less than a fraction $\gamma$ of $\bar{\mathcal{M}}$ with probability greater or equal to $b$ is given by

$$\nu_{\min} \approx \left(\frac{\sigma}{\mathcal{M}}\right)^2 \frac{2}{\gamma^2} \left[\mathrm{erf}^{-1}(b)\right]^2$$

where

$$\mathrm{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x \exp\left(-u^2\right) du. \tag{32}$$

Our experimental results show that for the same training task, different training algorithms yield significantly differing values of $(\sigma/\mathcal{M})^2$. Hence, to achieve a given level of statistical significance, different algorithms may require performing substantially differing numbers of learning trials.

Concerning generalization capability, the major issue is to decide which of two ANN classifiers is better than the other. This can be readily achieved by performing Student's $t$ test on the differences of the mean values of their classification accuracy distributions. By applying Student's $t$ test, we made sure in our OCR-related experiments that the number of learning trials was enough to ensure at least with 95% probability the truth of the hypothesis that ALECO is a better classifier than the other algorithms tried. To obtain statistically significant quantitative estimates of the degree of generalization improvement offered by the better algorithm, we could in principle apply formula 32 with $\mathcal{M}$ and $\sigma$ denoting the average and standard deviation of the classification accuracy distribution obtained by each algorithm. A small $\gamma$ would be required, however, to obtain very clearly separated averages for different algorithms, and this would lead to a prohibitively large number of learning trials.

### E. Selection of Learning Parameters

Each of the algorithms employed in this work has its own learning parameters, summarized here to establish notation.

- *BP (on/off-line):* The learning rate $\epsilon$ and the momentum factor $\alpha$ [6], [7], [60].
- *Quickprop:* The learning rate $\epsilon$, the momentum factor $\alpha$, the maximum growth factor $\mu$, and the weight decay term $\omega$ [8].
- *Delta-Bar-Delta:* The learning rate $\epsilon$, the learning rate increment $\kappa$, the learning rate proportion decrement $\varphi$, the base of the exponential average of the derivatives $\theta$, and the momentum factor $\alpha$ [29].
- *ALECO:* The parameters $\delta P$ and $\xi$ defined in Sections IV and V.

A fair comparison requires that the best learning parameter values be chosen for each algorithm and benchmark [23], [52]. Thus, learning parameters are adjusted, taking into account the guidelines of the algorithms' authors, to achieve the best possible performance concerning either avoidance of local minima and learning speed or generalization capability.

If learning speed and avoidance of local minima is evaluated, results are considered optimal when small minima and averages for the number of epochs needed to complete each benchmark task are obtained, subject to the condition that at least 70% of the experimental trials have passed the 0.4–0.6 convergence criterion. Both minima and averages can be important for learning speed evaluation. In the literature there exist two approaches, reported in [29] and [8], respectively. The first approach favors methods which give better values of $\mathcal{M}$ while the second, using the restart procedure, favors algorithms which give better minima of the number of epochs. Which of these is a better measure of learning speed performance depends heavily on the shape of the distribution of epochs needed to successfully complete a task. For example, it is clearly desirable that an algorithm exhibit a small $\sigma$ to $\mathcal{M}$ ratio; it is then reliable in its performance as regards learning speed and $\mathcal{M}$ is an adequate parameter by which to judge performance. On the other hand, if $\sigma/\mathcal{M}$ is of the order of $\mathcal{M}$ or larger, the distribution will extend far beyond its peak toward infinity and the peak can be closer to the minimum number of epochs than to $\mathcal{M}$. This minimum may then be considered as a better measure of learning speed, provided that the user of the algorithm is prepared to disregard the trouble caused by trials in which an excessive number of epochs is needed to achieve convergence.

If generalization capability is our prime concern (as in the OCR experiment), results are considered optimal when large minima and averages, but small standard deviations in the distributions of classification accuracy, are obtained. In the literature, not many reports pay due attention to the statistical nature of the results and the reliability of the classifiers. When classification accuracy distribution is considered (see, e.g., [17]), usually only its average is reported. The reliability of an FNN classifier, however, clearly depends on the standard deviation of its classification accuracy distribution.

## VII. Experimental Study: Results

### A. Presentation of the Results

Detailed experimental results on the performance of ALECO and the other supervised learning algorithms mentioned in Section VI-A are presented in Tables I–VIII. In particular, Tables I–IV refer to small-scale binary benchmarks, while Tables V–VII show results on large-scale binary benchmarks. Finally, Table VIII summarizes our generalization performance results regarding the OCR-related task.

In each table sufficient information is given which allows other researchers to reproduce our results (taking into account the factors common for all algorithms and benchmarks reported in Section VI-C). Thus, a brief benchmark description is given (benchmark identification, number of patterns and categories, reference where a full description can be found, if any), the architecture of the FNN which is called upon to solve the benchmark is described, the learning parameters used for each algorithm are reported, the maximum allowed number of epochs before declaration of failure is shown, and finally, the number of trials performed for each algorithm to ensure the minimum required level of statistical significance (Section VI-D) is given.

TABLE I
Experimental Results for the 4-5-5 Counter Problem

| Experiment | Type: COUNTER | | Categories: 5 | Input samples: 16 | Ref: [33] |
|---|---|---|---|---|---|
| MFNN system architecture | Input units: 4 | | hidden layers: 1 | hidden units: 5 | output units: 5 |
| Algorithms | ALECO | On BP | Off BP | Quickprop | Delta-Bar-Delta |
| Learning Parameters | $\delta P = 0.9$<br>$\xi = 0.75$ | $\epsilon = 0.50$<br>$\alpha = 0.70$ | $\epsilon = 0.50$<br>$\alpha = 0.5$ | $\epsilon = 3.00, \mu = 1.75$<br>$\omega = -0.0001, \alpha = 0.0$ | $\epsilon = 0.20, \kappa = 0.09$<br>$\varphi = 0.12, \theta = 0.70$<br>$\alpha = 0.70$ |
| Maximum allowed number of epochs | 3000 | | | | |
| Trials | 1000 | 1000 | 1000 | 1000 | 1000 |
| Successes (%) | 96.6 | 85.2 | 90.2 | 92.5 | 89.0 |
| Failures (%) | 3.4 | 14.8 | 9.8 | 7.5 | 11.0 |
| Distribution of epochs in successful trials | | | | | |
| Mean | 120.45 | 247.57 | 520.10 | 603.18 | 227.92 |
| Stdev | 140.5 | 76.25 | 504.28 | 517.51 | 241.56 |
| Maximum | 1516 | 992 | 2965 | 2921 | 2053 |
| Minimum | 49 | 155 | 148 | 78 | 53 |

TABLE II
Experimental Results for the 8-3-8 Encoder Problem

| Experiment | Type: ENCODER | | Categories: 8 | Input samples: 8 | Ref: [8] |
|---|---|---|---|---|---|
| MFNN system architecture | Input units: 8 | | hidden layers: 1 | hidden units: 3 | output units: 8 |
| Algorithms | ALECO | On BP | Off BP | Quickprop | Delta-Bar-Delta |
| Learning Parameters | $\delta P = 2.00$<br>$\xi = 0.75$ | $\epsilon = 2.50$<br>$\alpha = 0.50$ | $\epsilon = 1.50$<br>$\alpha = 0.50$ | $\epsilon = 5.00, \mu = 1.75$<br>$\omega = -0.0001, \alpha = 0.0$ | $\epsilon = 0.50, \kappa = 0.45$<br>$\varphi = 0.15, \theta = 0.40$<br>$\alpha = 0.80$ |
| Maximum allowed number of epochs | 1000 | | | | |
| Trials | 1000 | 1000 | 1000 | 1000 | 1000 |
| Successes (%) | 100.0 | 100.0 | 100.0 | 100.0 | 99.9 |
| Failures (%) | 0.0 | 0.0 | 0.0 | 0.0 | 0.1 |
| Distribution of epochs in successful trials | | | | | |
| Mean | 36.80 | 172.06 | 145.77 | 83.26 | 68.34 |
| Stdev | 14.39 | 79.50 | 53.62 | 97.51 | 64.08 |
| Maximum | 127 | 517 | 370 | 946 | 941 |
| Minimum | 17 | 25 | 56 | 16 | 25 |

In reporting our results, we have tried to be as informative as possible, giving detailed information about the distribution of the quantity of interest (number of epochs or classification accuracy). Detailed results are shown including the percentage of successful and unsuccessful trials; the mean, standard deviation, minimum, and maximum of the epoch distribution in successful learning trials in the binary benchmarks, where learning speed is the major objective (Tables I–VII); and the mean, standard deviation, minimum, and maximum of the classification accuracy distribution in successful and failed learning trials in the case of the OCR-related experiment (Table VIII).

Our presentation of learning speed results implies that the epoch has been chosen as an appropriate unit by which to measure learning time. In principle, learning speed can be

TABLE III
EXPERIMENTAL RESULTS FOR THE 6-6-1 MULTIPLEXER PROBLEM

| Experiment | Type: MULTIPLEXER | | Categories: 2 | Input samples: 64 | Ref: [29] |
|---|---|---|---|---|---|
| AFSE system architecture | Input units: 6 | | Hidden layers: 1 | Hidden units: 6 | output units: 1 |
| Algorithms | ALECO | On BP | Off BP | Quickprop | Delta-Bar-Delta |
| Learning Parameters | $\delta P = 1.20$<br>$\xi = 0.85$ | $\epsilon = 1.50$<br>$\alpha = 0.0$ | $\epsilon = 0.30$<br>$\alpha = 0.80$ | $\epsilon = 0.5, \mu = 2.5$<br>$\omega = -0.0001, \alpha = 0.0$ | $\epsilon = 0.20, \kappa = 0.09$<br>$\varphi = 0.12, \theta = 0.50$<br>$\alpha = 0.80$ |
| Maximum allowed number of epochs | 2000 | | | | |
| Trials | 1000 | 1000 | 1000 | 1000 | 2000 |
| Successes (%) | 99.7 | 99.6 | 84.0 | 96.1 | 82.1 |
| Failures (%) | 0.3 | 0.4 | 16.0 | 3.9 | 17.9 |
| Distribution of epochs in successful trials | | | | | |
| Mean | 63.56 | 166.47 | 124.33 | 236.04 | 135.76 |
| Stdv | 47.09 | 74.74 | 114.66 | 230.60 | 204.72 |
| Maximum | 665 | 1145 | 996 | 1968 | 1959 |
| Minimum | 25 | 86 | 41 | 48 | 28 |

TABLE IV
EXPERIMENTAL RESULTS FOR THE XOR PROBLEM

| Experiment | Type: XOR | | Categories: 2 | Input samples: 4 | Ref: [66] |
|---|---|---|---|---|---|
| AFSE system architecture | Input units: 2 | | Hidden layers: 1 | Hidden units: 2 | output units: 1 |
| Algorithms | ALECO | On BP | Off BP | Quickprop | Delta-Bar-Delta |
| Learning Parameters | $\delta P = 0.65$<br>$\xi = 0.80$ | $\epsilon = 0.90$<br>$\alpha = 0.70$ | $\epsilon = 4.50$<br>$\alpha = 0.80$ | $\epsilon = 2.00, \mu = 1.20$<br>$\omega = -0.0001, \alpha = 0.0$ | $\epsilon = 7.00, \kappa = 0.25$<br>$\varphi = 0.12, \theta = 0.70$<br>$\alpha = 0.80$ |
| Maximum allowed number of epochs | 2000 | | | | |
| Trials | 1000 | 2000 | 1000 | 5000 | 1000 |
| Successes (%) | 98.30 | 73.50 | 91.90 | 80.20 | 92.70 |
| Failures (%) | 1.70 | 26.50 | 8.10 | 19.80 | 7.30 |
| Distribution of epochs in successful trials | | | | | |
| Mean | 38.05 | 175.12 | 73.88 | 130.60 | 66.02 |
| Stdv | 29.94 | 215.96 | 79.16 | 272.95 | 72.59 |
| Maximum | 483 | 1978 | 1176 | 1999 | 1245 |
| Minimum | 13 | 24 | 25 | 26 | 26 |

TABLE V
EXPERIMENTAL RESULTS FOR THE 128-7-128 ENCODER PROBLEM

| Experiment | Type: ENCODER | | Categories: 128 | Input samples: 128 | Ref: [8] |
|---|---|---|---|---|---|
| AFSE system architecture | Input units: 128 | | Hidden layers: 1 | Hidden units: 7 | output units: 128 |
| Algorithms | ALECO | On BP | Off BP | Quickprop | Delta-Bar-Delta |
| Learning Parameters | $\delta P = 1.80$<br>$\xi = 0.50$ | $\epsilon = 0.50$<br>$\alpha = 0.70$ | No convergence | No convergence | No convergence |
| Maximum allowed number of epochs | 1000 | | | | |
| Trials | 100 | 100 | 100 | 100 | 100 |
| Successes (%) | 100.0 | 99.00 | 0.0 | 0.0 | 0.0 |
| Failures (%) | 0.0 | 1.00 | 100.0 | 100.0 | 100.0 |
| Distribution of epochs in successful trials | | | | | |
| Mean | 171.01 | 340.13 | | | |
| Stdv | 9.59 | 43.49 | | | |
| Maximum | 196 | 477 | | | |
| Minimum | 150 | 264 | | | |

TABLE VI
EXPERIMENTAL RESULTS FOR THE 256-8-256 ENCODER PROBLEM

| Experiment | Type: ENCODER | | Categories: 256 | Input samples: 256 | Ref: [8] |
|---|---|---|---|---|---|
| AFSE system architecture | Input units: 256 | | Hidden layers: 1 | Hidden units: 8 | output units: 256 |
| Algorithms | ALECO | On BP | Off BP | Quickprop | Delta-Bar-Delta |
| Learning Parameters | $\delta P = 2.0$<br>$\xi = 0.50$ | $\epsilon = 0.40$<br>$\alpha = 0.60$ | No convergence | No convergence | No convergence |
| Maximum allowed number of epochs | 1000 | | | | |
| Trials | 50 | 50 | 50 | 50 | 50 |
| Successes (%) | 100.0 | 100.0 | 0.0 | 0.0 | 0.0 |
| Failures (%) | 0.0 | 0.0 | 100.0 | 100.0 | 100.0 |
| Distribution of epochs in successful trials | | | | | |
| Mean | 224.04 | 476.68 | | | |
| Stdv | 9.93 | 89.96 | | | |
| Maximum | 256 | 699 | | | |
| Minimum | 201 | 358 | | | |

valuated using either the concept of epoch, which is widely accepted, or similar ones [8], or the concept of computational complexity as partially involved in [63]. In terms of complexity in the weight update calculations involved in one epoch, our experiments verified that there are no great differences between the algorithms tested. We believe that the epoch, apart from being a convenient and generally accepted unit for measuring learning time [8], is also compatible with the concept of the "100-step program" constraint [64] extended in the training procedure. In biological learning, the number of "training set" presentations involved in a learning task is a small number rather than a number of the order of thousands. Moreover, we can assume that whenever exponential time is involved in computations within the same epoch, neural networks (the biological ones, and also the models we investigate) may be able to provide speed at the cost of an excessive network size [65]. For these reasons we have adopted the epoch as the unit to be used in our study.

### B. Discussion of the Results

Evidently, ALECO outperforms off-line BP in all four small-scale benchmarks in terms of learning speed and avoidance of local minima. Ratios of the average number of epochs needed to solve the tasks using the two algorithms range from 4.32 in the 4-5-5 counter problem to 1.94 in the XOR problem, always in favor of ALECO. The corresponding ratios for the minimum number of epochs needed to solve these tasks range from 3.29 in the 8-3-8 encoder to 1.64 in the 6-6-1 multiplexer. Moreover, our method also achieves much faster learning than on-line BP, Quickprop, and Delta-Bar-Delta: Much better averages are obtained than these three algorithms for all benchmarks; the minimum number of epochs is much better than that achieved by Delta-Bar-Delta and on-line BP in all cases and better or, at worst, comparable to that achieved by Quickprop. Compared to the other methods as regards the percentage of successful learning trials, ALECO clearly exhibits improved performance. It outperforms all algorithms

TABLE VII
EXPERIMENTAL RESULTS FOR THE 11-11-1 MULTIPLEXER PROBLEM

| Experiment | Type: MULTIPLEXER | | Categories: 2 | Input samples: 2048 | Ref: [29] |
|---|---|---|---|---|---|
| MFN system architecture | Input units: 11 | | Hidden layers: 1 | Hidden units: 11 | output units: 1 |
| Algorithm | ALECO | On BP | Off BP | Quickprop | Delta-Bar-Delta |
| Learning Parameters | $\delta P = 0.65$ $\xi = 0.50$ | $\epsilon = 0.60$ $\alpha = 0.70$ | No convergence | No convergence | No convergence |
| Maximum allowed number of epochs | 1000 | | | | |
| Trials | 100 | 100 | 100 | 100 | 100 |
| Successes (%) | 100.0 | 94.0 | 0.0 | 0.0 | 0.0 |
| Failures (%) | 0.0 | 6.0 | 100.0 | 100.0 | 100.0 |
| Distribution of epochs in successful trials | | | | | |
| Mean | 138.47 | 279.68 | | | |
| Std. dev. | 36.87 | 91.68 | | | |
| Maximum | 305 | 548 | | | |
| Minimum | 79 | 160 | | | |

in all cases, except in the encoder problem where no failures were encountered by all algorithms.

Results in the three large-scale benchmarks can be summarized as follows:

- Concerning convergence ability, we notice that off-line BP, Quickprop, and Delta-Bar-Delta cannot converge at all in the benchmarks tried within the specified limit of epochs until failure. By contrast, on-line BP and ALECO exhibit good convergence ability in all three tasks tried, with ALECO slightly outperforming on-line BP in the 11-11-1 multiplexer problem.
- Concerning learning speed, measured either by the average or the minimum of the distribution of epochs needed to successfully complete a task, ALECO clearly outperforms its closest rival (on-line BP) in all tasks by, approximately, a factor of two.
- ALECO exhibits a relatively small standard deviation in the distribution of epochs needed to successfully complete a task, thus exhibiting reliability of performance as regards learning speed.

These results are in compliance with work by Fogelman Soulie [14] reporting that on-line BP exhibits very good learning abilities in large-scale problems. Moreover, they demonstrate the emergence of an excellent new training algorithm, ALECO, for large-scale networks and problems.

Results in the OCR-related task can be summarized as follows:

- Concerning classification accuracy and reliability in generalization performance, we notice that ALECO outperforms both on-line BP and Quickprop, while off-line BP and Delta-Bar-Delta give insignificant generalization results. More specifically, we notice that ALECO improves the mean classification error of on-line BP by a factor ranging from 1.12 to 2.2, and the mean classification error of Quickprop by a factor ranging from 1.15 to 2.56. Moreover, ALECO is much better than the other algorithms not only exhibiting large minima in its classification accuracy distribution but also small standard deviations. It is thus a more reliable classifier.
- Concerning convergence ability, we notice that off-line BP and Delta-Bar-Delta cannot converge at all in this benchmark, within the specified limit of epochs until failure, while Quickprop converges only in the 2% of the trials tried (although it is capable of learning most patterns, as reflected in its acceptable classification accuracy). By contrast, on-line BP and ALECO exhibit very good convergence ability.
- Concerning learning speed, measured by the average of the distribution of epochs needed to successfully complete this task, ALECO outperforms on-line BP by a factor of 1.15, even though learning parameters were chosen to ensure optimal generalization rather than learning speed (on-line BP solves the problem in 120 epochs on average, while ALECO solves it in 105 epochs).

Practical guidelines can be given for selecting optimal values for the learning parameters $\delta P$ and $\xi$: For all small-scale benchmarks, similar performances were recorded with $0.5 < \xi < 0.9$ and $1.0 < \delta P < 2.0$, indicating that results are not very sensitive to the exact values of the parameters. Following the example of Jacobs [29], we performed additional runs of ALECO using common values ($\delta P = 1.5$ and $\xi = 0.85$) for all four benchmarks. Deterioration of the mean number of epochs, compared to the optimal values shown in Tables I–IV, was never more than 30%. Larger-scale problems are more sensitive to the selection of $\delta P$, but not to the selection of $\xi$ for which a value around 0.5 works well for all benchmarks.

## VIII. CONCLUSION

In this paper, an efficient training algorithm for FNN was proposed incorporating suitable construction of internal representations and momentum acceleration in its formalism and exhibiting the following attractive features:

- Solid theoretical background, based on rigorous nonlinear programming techniques.
- Proved convergence to global or local minima of the mean square error cost function for small enough learning step. This property is shared with the BP algorithm, but not with some of its descendants of heuristic origin.
- Faster learning than the BP algorithm, from which it is inspired, and from other reputedly fast training algorithms.
- Efficiency in avoiding local minima.
- Good scalability properties.
- Improved generalization capability.

To confirm the existence of such properties in FNN, when they are trained using ALECO, experiments involving binary benchmarks of small and large scale, as well as a large-scale OCR-related problem, were carefully designed and carried out. ALECO was found to outperform BP and some of its reputedly successful descendants regarding important learning properties, viz. learning speed, avoidance of local minima and generalization capability.

Clearly, ALECO represents the first step in a long-term program of research on the problem of incorporating different kinds of information in FNN training algorithms using rigorous

TABLE VIII
EXPERIMENTAL RESULTS FOR THE OCR PROBLEM DEFINED IN THE TEXT

| Experiment | Type: OCR | | Categories: 32 | Input samples: 960 | Ref: - |
|---|---|---|---|---|---|
| MFNN system architecture | input units: 41 | | hidden layers: 1 | hidden units: 40 | output units: 32 |
| Algorithms | ALECO | On BP | Off BP | Quickprop | Delta-Bar-Delta |
| Learning Parameters | $\delta P = 0.4$<br>$\xi = 0.4$ | $\epsilon = 0.4$<br>$\alpha = 0.5$ | No convergence<br>Generalization $\ll 10\%$ | $\epsilon = 0.4$<br>$\alpha = 0.5$ | No convergence<br>Generalization $\ll 10\%$ |
| Maximum allowed number of epochs | 500 | | | | |
| Trials | 50 | 50 | 50 | 50 | 50 |
| Successes (%) | 100.0 | 100.0 | 0.0 | 2.0 | 0.0 |
| Failures (%) | 0.0 | 0.0 | 100.0 | 98.0 | 100.0 |
| Mean Accuracy/Mean Classification Error % | | | | | |
| Courier | 80.14/19.86 | 77.79/22.21 | | 77.05/22.95 | |
| Text-1 | 99.34/0.66 | 99.19/0.81 | | 98.99/1.01 | |
| Text-2 | 99.15/0.85 | 98.48/1.52 | | 97.77/2.23 | |
| Text-3 | 99.76/0.24 | 99.47/0.53 | | 99.59/0.41 | |
| Stdv % | | | | | |
| Courier | 1.40 | 1.60 | | 3.27 | |
| Text-1 | 0.29 | 0.40 | | 0.40 | |
| Text-2 | 0.26 | 0.44 | | 1.00 | |
| Text-3 | 0.12 | 0.37 | | 0.23 | |
| Min accuracy/Max classification error % | | | | | |
| Courier | 76.50/23.50 | 74.06/25.94 | | 67.81/32.19 | |
| Text-1 | 98.64/1.36 | 97.49/2.51 | | 97.87/2.13 | |
| Text-2 | 98.10/1.90 | 97.21/2.79 | | 92.70/7.30 | |
| Text-3 | 99.37/0.63 | 97.28/2.72 | | 98.80/1.20 | |
| Max classification accuracy % | | | | | |
| Courier | 82.95 | 81.25 | | 83.50 | |
| Text-1 | 99.90 | 99.87 | | 99.74 | |
| Text-2 | 99.54 | 99.07 | | 99.20 | |
| Text-3 | 99.94 | 99.87 | | 99.90 | |

constrained optimization techniques. It is the concerted incorporation of such detailed information into the same algorithm which will hopefully eliminate the need for heuristics and lead to increasingly efficient FNN training schemes.

## APPENDIX

Let us denote by $e_i$ the value of the stochastic variable in consideration (e.g., number of epochs needed to successfully complete a task), recorded in the $i$th successful trial. The expected value of any of the $e_i$ represents the "true" average $\bar{\mathcal{M}}$ of the stochastic variable distribution. Moreover, let $\Sigma$ denote the standard deviation of any of the $e_i$. According to the central limit theorem, the distribution of the sample average $\mathcal{M} = (e_1 + e_2 + \cdots + e_\nu)/\nu$ for a sufficiently large number $\nu$ of independent trials tends to the normal distribution $N(\bar{\mathcal{M}}, \Sigma/\sqrt{\nu})$. It follows that $(\mathcal{M} - \bar{\mathcal{M}})\sqrt{\nu}\Sigma^{-1}$ tends to the normal distribution $N(0,1)$. It is required that the probability

$$\mathcal{P}\left[|\mathcal{M} - \bar{\mathcal{M}}| \le \gamma\bar{\mathcal{M}}\right] = \mathcal{P}\left[|\mathcal{M} - \bar{\mathcal{M}}|\sqrt{\nu}\Sigma^{-1} \le \gamma\bar{\mathcal{M}}\sqrt{\nu}\Sigma^{-1}\right] \quad (33)$$

be greater or equal to $b$. Therefore

$$\frac{2}{\sqrt{2\pi}} \int_0^{\gamma\bar{\mathcal{M}}\sqrt{\nu}\Sigma^{-1}} \exp(-t^2/2)\, dt \ge b \quad (34)$$

which can be rewritten as

$$\operatorname{erf}\left(\frac{\gamma\bar{\mathcal{M}}\sqrt{\nu}}{\sqrt{2}\Sigma}\right) \ge b. \quad (35)$$

By substituting $\bar{\mathcal{M}}$ and $\Sigma$ by their experimentally determined estimates $\mathcal{M}$ and $\sigma$, we readily obtain the experimental estimate given by (32) for the minimum number $\nu_{\min}$ of trials required to achieve the desired level of statistical significance for $\mathcal{M}$.

## REFERENCES


[1] A. N. Kolmogorov, "On the representation of continuous functions of many variables by superposition of continuous functions of one variable and addition," *Doklady Akademii Nauk SSSR*, vol. 144, pp. 679–681, 1957. Amer. Math. Soc. translation, vol. 28, pp. 55–59, 1963.

[2] D. A. Sprecher, "On the structure of continuous functions of several variables," *Trans. Amer. Math. Soc.*, vol. 115, pp. 340–355, 1965.

[3] R. Hecht-Nielsen, "Kolmogorov mapping neural network existence theorem," in *Proc. IEEE 1st Int. Conf. Neural Networks*, San Diego, 1987, pp. III11–III13.

[4] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks*, vol. 2, pp. 359–366, 1989.

[5] K. Funahashi, "On the approximate realization of continuous mappings by neural networks," *Neural Networks*, vol. 2, pp. 183–192, 1989.

[6] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," in *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, D. E. Rumelhart

and J. L. McClelland, Eds. Cambridge, MA: MIT Press, ch. 8, 1986, pp. 318–362.
[7] ——, "Learning representations by backpropagating errors," *Nature*, vol. 323, pp. 533–536, 1986.
[8] S. E. Fahlman, "Faster learning variations on backpropagation: An empirical study," in *Proc. 1988 Connectionist Models Summer School*, 1988, pp. 38–51.
[9] S. Mori, C. Y. Suen, and K. Yamamoto, "Historical review of OCR research and development," *Proc. IEEE*, vol. 80, no. 7, pp. 1029–1058, 1992.
[10] Y. le Cun, "Generalization and network design strategies," in *Connectionism in Perspective*. Amsterdam, The Netherlands: Elsevier, 1989.
[11] F. Crick, "The recent excitement about neural networks," *Nature*, vol. 337, pp. 129–132, 1989.
[12] R. Fletcher, *Practical Methods of Optimization*, vol. 1. New York: Wiley, 1980.
[13] A. A. Goldstein, "On steepest descent," *J. SIAM Contr. Ser. A*, vol. 3, no. 1, pp. 147–151, 1965.
[14] F. Fogelman Soulie, "Neural network architectures and algorithms: A perspective," in *Artificial Neural Networks*, T. Kohonen, K. Makisara, O. Simula, and J. Kangas, Eds. Amsterdam, The Netherlands: Elsevier, 1991, pp. 605–615.
[15] A. Khotanzad and J. H. Lu, "Distortion invariant character recognition by a multilayer perceptron and backpropagation learning," in *IEEE 2nd Int. Conf. Neural Networks*, 1988, pp. 625–632.
[16] Y. Lee, S. H. Oh, and M. W. Kim, "An analysis of premature saturation in backpropagation learning," *Neural Networks*, vol. 6, pp. 719–728, 1993.
[17] L. F. Wessels and E. Barnard, "Avoiding false local minima by proper initialization of connections," *IEEE Trans. Neural Networks*, vol. 3, no. 6, pp. 899–905, 1992.
[18] A. Blum and R. Rivest, "Training a three-node neural network is np-complete," in *Proc. Computa. Learning Theory (COLT) Conf.*, 1988, pp. 9–18.
[19] S. Judd, "On the complexity of loading shallow neural networks," *J. Complexity*, vol. 4, pp. 177–192, 1988.
[20] G. Tesauro and Y. H. Ahmad, "Asymptotic convergence of backpropagation," *Neural Computa.*, vol. 1, no. 3, pp. 382–391, 1989.
[21] J. A. Benediktsson, P. H. Swain, and O. K. Ersoy, "Neural network approches versus statistical methods in classification of multisource remote sensing data," *IEEE Trans. Geosci. Remote Sensing*, vol. 28, no. 4, pp. 540–552, 1990.
[22] H. Bischof, W. Schneider, and A. Pinz, "Multispectral classification of landsat-images using neural networks," *IEEE Trans. Geosci. Remote Sensing*, vol. 30, no. 3, pp. 482–490, 1992.
[23] A. van Ooyen and B. Nienhuis, "Improving the convergence of the backpropagation algorithm," *Neural Networks*, vol. 5, pp. 465–471, 1992.
[24] D. B. Parker, "Optimal algorithms for adaptive networks: Second-order backpropagation, second-order direct propagation, and second-order Hebbian learning," in *Proc. IEEE 1st Int. Conf. Neural Networks*, San Diego, 1987, pp. 593–600.
[25] S. Becker and Y. le Cun, "Improving the convergence of backpropagation learning with second-order methods," in *Proc. Connectionist Models Summer School*, Pittsburgh, PA, 1988, pp. 29–37.
[26] A. H. Kramer and Sangiovanni-Vincentelli, "Efficient parallel learning algorithms for neural networks," in *Advances in Neural Information Processing Systems*, D. S. Touretzky, Ed. San Mateo, CA: Morgan Kaufmann, 1988, pp. 40–48.
[27] R. L. Watrous, "Learning algorithms for connectionist networks: Applied gradient methods of nonlinear optimization," in *Proc. IEEE 1st Int. Conf. Neural Networks*, San Diego, pp. II619–II627, 1987.
[28] S. Singhal and L. Wu, "Training feedforward networks with the extended Kalman filter," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing*, Glasgow, Scotland, 1989, pp. 1187–1190.
[29] R. A. Jacobs, "Increased rates of convergence through learning rate adaptation," *Neural Networks*, vol. 1, pp. 295–307, 1988.
[30] A. Krogh, G. I. Thorbergsson, and J. A. Hertz, "A cost function for internal representations," in *Advances in Neural Information Processing Systems*, D. S. Touretzky, Ed. San Mateo, CA: Morgan Kaufmann, 1990, pp. 733–740.
[31] P. J. Werbos, "Backpropagation through time: what it does and how to do it," in *Proc. IEEE*, C. Lau and B. Widrow, Eds., pp. 1550–1560, 1990.
[32] Y. le Cun, L. D. Jackel, B. E. Boser, J. S. Denker, H. Graf, I. Guyon, D. Henderson, R. E. Howard, and W. Hubbard, "Handwritten digit recognition: Applications of neural network chips and automatic learning," *IEEE Communications Mag.*, pp. 41–46, Nov. 1989.
[33] T. J. Sejnowski and C. Rosenberg, "Parallel networks that learn to pronounce English text," *Complex Syst.*, vol. 1, pp. 145–168, 1987.
[34] R. P. Gorman and T. J. Sejnowski, "Analysis of hidden units in a layered network trained to classify sonar targets," *Neural Networks*, vol. 1, pp. 75–89, 1988.
[35] R. Rohwer, "The 'moving targets' training algorithm," in *Advances in Neural Information Processing Systems*, D. S. Touretzky, Ed. San Mateo, CA: Morgan Kaufmann, 1990, pp. 558–565.
[36] T. Grossman, "The CHIR algorithm for feed forward networks with binary weights," in *Advances in Neural Information Processing Systems*, D. S. Touretzky, Ed. San Mateo, CA: Morgan Kaufmann, 1990, pp. 516–523.
[37] T. Grossman, R. Meir, and E. Domany, "Learning by choice of internal representations," in *Advances in Neural Information Processing Systems*, D. S. Touretzky, Ed. San Mateo, CA: Morgan Kaufmann, 1990, pp. 73–80.
[38] Y. Chauvin, "A backpropagation algorithm with optimal use of hidden units," in *Advances in Neural Information Processing Systems*, D. S. Touretzky, Ed. San Mateo, CA: Morgan Kaufmann, 1989, pp. 519–526.
[39] J. Denker, D. Schwartz, D. Wittner, S. Solla, R. Howard, L. Jackel, and J. Hopfield, "Large automatic learning, rule extraction and generalization," *Complex Syst.*, vol. 1, pp. 877–922, 1987.
[40] A. S. Weigend, D. E. Rumelhart, and B. A. Huberman, "Generalization by weight elimination with application to forecasting," in *Advances in Neural Information Processing Systems*, D. S. Touretzky, Ed. San Mateo, CA: Morgan Kaufmann, 1991, pp. 875–882.
[41] H. Drucker and Y. le Cun, "Improving generalization performance using double backpropagation," *IEEE Trans. Neural Networks*, vol. 3, no. 6, pp. 991–997, 1992.
[42] S. S. Rao, *Optimization Theory and Applications*. New Delhi, India: Wiley Eastern, 1984.
[43] A. E. Bryson and W. F. Denham, "A steepest-ascent method for solving optimum programming problems," *J. Appl. Mechanics*, vol. 29, pp. 247–257, 1962.
[44] S. E. Fahlman and C. Lebiere, "The cascade-correlation learning architecture," in *Advances in Neural Information Processing Systems*, D. S. Touretzky, Ed. San Mateo, CA: Morgan Kaufmann, pp. 524–532, 1990.
[45] L. F. Wessels, E. Barnard, and E. Rooyen, "The physical correlates of local minima," in *Proc. Int. Neural Networks Conf.*, Paris, 1990, p. 985.
[46] R. S. Sutton, "Two problems with backpropagation and other steepest-descent learning procedures for networks," in *Proc. 8th Annu. Conf. Cognitive Sci. Soc.*, 1986, pp. 823–831.
[47] D. R. Hush, B. Horne, and J. M. Salas, "Error surfaces for multilayer perceptrons," *IEEE Trans. Syst., Man Cybern.*, vol. 22, no. 5, pp. 1152–1161, 1992.
[48] S. J. Perantonis and D. A. Karras, "An efficient constrained learning algorithm with momentum acceleration," *Neural Networks*, vol. 8, no. 2, pp. 237–249, 1994.
[49] J. Moody, "Note on generalization, regularization and architecture selection in nonlinear learning systems," in *Neural Networks for Signal Processing*, B. H. Juang, S. Y. Kung, and C. A. Kamm, Eds. Piscataway, NJ: IEEE Press, 1991.
[50] Y. le Cun, "A theoretical framework for backpropagation," in *Proc. Connectionist Models Summer School*, Pittsburgh, PA, 1988, pp. 21–28.
[51] W. S. Cheung and J. K. Hammond, "On a generalized backpropagation algorithm based on optimal control theory," in *Proc. Int. Joint Conf. Neural Networks*, Singapore, 1991, pp. 821–826.
[52] T. Tollenaere, "Super SAB: Fast adaptive back propagation with good scaling properties," *Neural Networks*, vol. 3, pp. 561–573, 1990.
[53] S. J. Perantonis and D. A. Karras, "An integrated environment for experimenting with feedforward neural networks," NCSR "Demokritos," Tech. Rep., 1994.
[54] J. Schmidhuber, "Accelerated learning in backpropagation nets," Institut für Informatik, Technische Universität München, West Germany, Tech. Rep., 1988.
[55] B. D. Ripley, "Flexible nonlinear approaches to classification," in *From Statistics to Neural Networks*, NATO ASI series, 1994.
[56] M. Minsky and S. Papert, *Perceptrons*. Cambridge, MA: MIT Press, 1969.
[57] P. J. Burt and E. H. Adelson, "The Laplacian pyramid as a compact image code," *IEEE Trans. Commun.*, vol. 31, no. 4, pp. 532–540, 1983.
[58] P. J. Burt, "Fast filter transforms for image processing," *Comput. Graphics Image Process.*, vol. 16, pp. 20–51, 1981.
[59] H. Y. Y. Sanossian and D. J. Evans, "An acceleration method for the backpropagation learning algorithm," in *Proc. 4th Int. Conf. Neural Networks Applicat., Neuro-nimes '91*, Nimes, France, 1991, pp. 377–385.

[60] Y. Pao, *Adaptive Pattern Recognition and Neural Networks*. Reading, MA: Addison-Wesley, 1989.
[61] O. Kufudaki and J. Horejs, "PAB-parameters adapting backpropagation," *Neural Network World*, vol. 1, no. 5, pp. 267–274, 1991.
[62] G. Bebis, G. Papadourakis, and M. Georgiopoulos, "Backpropagation: Increasing rate of convergence by predictable pattern loading," *Intell. Syst. Rev.*, vol. 1, no. 3, 1989.
[63] S. Shah, F. Palmieri, and M. Datum, "Optimal filtering algorithms for fast learning in feedforward neural networks," *Neural Networks*, vol. 5, pp. 779–787, 1992.
[64] J. A. Feldman, "Connectionist models and their applications: introduction," *Cognitive Scie.*, vol. 9, pp. 1–2, 1985.
[65] Y. S. Abu-Mostafa, "Neural networks for computing?," in *Proc. AIP Conf. 151*, Snowbird, UT, 1986, pp. 1–6.

**Dimitris A. Karras** (A'86) received the diploma in electrical engineering in 1985 and the Ph.D. degree in computer science (neural networks) in 1995, both from National Technical University of Athens Electrical Engineering Department. He also received the B.Sc. degree in mathematics from the University of Athens, Greece, in 1990.

He is currently a Collaborating Research Scientist at the Neural Network Laboratory, Institute of Informatics of Telecommunications, NCSR "Demokritos," Athens, Greece. His main research interests include neural networks, pattern recognition, image and signal processing, and nonlinear systems theory. He has participated in two Greek and EU research programs. He is the author or coauthor of more than 25 scientific papers, published in journals and international conference proceedings, in the fields of neural networks and signal processing.

**Stavros J. Perantonis** received the B.Sc. degree in physics from the University of Athens, Greece, in 1984, the D.Phil. degree in theoretical physics from the University of Oxford, England, in 1987, and the M.Sc. degree in Computer Science from the University of Liverpool, England, in 1990.

From 1987 to 1990 he was a postdoctoral Research Associate at the Department of Applied Mathematics and Theoretical Physics, University of Liverpool, supported by a SERC fellowship. He is currently an Associate Researcher at the Artificial Neural Networks Laboratory, Institute of Informatics and Telecommunications, National Center for Scientific Research "Demokritos," Athens, Greece. His research interests include the theory of artificial neural networks and their applications, with emphasis on pattern recognition, image processing, and prediction. He has been involved in several national and European research projects and has authored or coauthored 40 papers published in scientific journals, books, and international conference proceedings.