FAULT TOLERANCE IN NEURAL NETWORKS: THEORETICAL ANALYSIS AND SIMULATION RESULTS

Vincenzo PIURI, Mariagiovanna SAMI, Renato STEFANELLI
Department of Electronics, Politecnico di Milano
piazza L. da Vinci 32, 20133 Milano, Italy

ABSTRACT
The problem of fault-tolerance in relation with neural nets is presently the subject of much research; while most authors deal with aspects related to specific VLSI implementations, work is going on also on the intrinsic capacity of survival to faults characterizing neural systems. In the present paper, we deal with this second theme, considering in particular multi-layered feed-forward nets. The study is performed on the abstract neural graphs, thus involving errors rather than faults; after an initial analysis of the error model, the effects of errors are mathematically derived and the conditions allowing to achieve complete recovery from faults through re-distribution of weights in the network (or otherwise allowing to grant pre-determined upper bounds on errors) are derived. Simulation results are then presented identifying the effect of such errors onto the neural computation. It is seen that (unless a good measure of redundancy is present in the net from the beginning) even single errors affect in a relevant way the computation; correction of this effect is sought through repeated learning, i.e., an operation leading to the weight adjustment previously discussed in theoretical terms. The results of such policy are also analyzed; again, it will be seen that even in this case errors affect the net's operation unless the net structure is initially redundant.

INTRODUCTION
Artificial Neural Networks (ANNs) are nowadays considered more than a promising solution for the increasing demand of massive computation in a large class of applications (e.g. signal and image processing). Advances in integration technologies and architectural design allow in fact actual implementations at reasonable costs.

The possibility of implementing ANNs by VLSI or WSI technologies and their application in mission-critical areas inevitably lead to considering the associated problems of defect and fault tolerance. This should be examined first of all with reference to the intrinsic mode of operation of such system, independently of the particular implementation technology adopted. In an ANN, computation and information are not localised, but they can be considered holographically distributed in the system. An error in computation by a single neuron or in one synaptic weight affects the whole computation, although in general it does not completely destroy any part of it. Moreover, through its learning capabilities, an ANN may achieve redistribution of information and computation by adjusting the activities of fault-free processing units and by minimising the influence of the faulty units upon the final results.

Several authors have dealt up to now with aspects of defect-and fault-tolerance relating to specific implementations of neural nets. Thus, for example, in [1] behavioral fault models are defined with reference to multi-layered nets implemented by analog systems, and the impact of the physical faults is examined at various abstraction levels. Following the same line, in [2] a relationship is identified between physical defects and failures and the maximum size of the device, again corresponding to a given analog implementation approach. In [3], on the other hand, a specific digital architecture supporting the mapping of a number of neural nets is envisioned and solutions allowing it to support defect and fault tolerance are discussed: the approach actually refers rather to the supporting architecture (in the specific instance, a rectangular array) than to the characteristics of the neural net mapped onto it.

In the present paper, we consider on the contrary the abstract graph model of a class of neural nets — namely, multi-layered feed-forward nets — and analyze first of all the behavioral error model that can be derived for them, independently of the implementation. Given such errors, we discuss their influence on the results of neural computation. It is seen that in general the effects cannot be a-priori quantified and are propagated throughout the system (in a way that, among other points, makes it impossible to adopt any behavioral testing solution for diagnosis of the system). Complete fault-tolerance requires that some sort of re-configuration of the network or of its parameters allow to nullify such errors; in the case of an ANN, such reconfiguration is first of all thought in terms of a new distribution of weight values through the fault-free neurons. Conditions granting this type of fault-tolerance are mathematically derived.

The theoretical results achieved in this first section are then checked by extensive recourse to simulation. The results of such phase allow to state first of all that — contrary to assertions of some researchers in the field — errors present in neural nets do affect, and in a relevant way, the computation capacity of the nets. The classification performed by the net is always affected by presence of a faulty neuron, even when the initial network was redundant with respect to the "minimal structure" capable of performing the required classification; only a subsequent "renewed learning" phase may lead it to overcome (at least partially) the effects of the error.

To achieve meaningful statistics, different structures of multi-layered nets are considered, all supporting recognition of a given set of patterns but characterized by varying numbers of neurons, and by different distributions of neurons among layers. In fact, redundancy is considered not only in terms of neurons but also in terms of synapses (different distributions of the same number of neurons will involve different numbers of synapses).
THE ERROR MODEL
AND ITS THEORETICAL IMPLICATIONS

The network model we refer to is the classical multi-layered perceptron proposed by Rosenblatt (see [4]) in which operation of any single neuron $n_i$ is given by:

$$z_i = f\left(\sum_j w_{ij} x_j - \theta_i\right)$$  \hspace{1cm} (1)

As a consequence, the individual neuron can be modelled as in figure 1, where the synaptic weight $w_{ij}$ together with the multiplication by input signal $x_j$ are associated with the synapsis connecting neuron $n_j$ with neuron $n_i$, while the summation over all input signals to $z_i$ and the subsequent neural nonlinear evaluation function are represented by individual corresponding operators associated with neuron $n_i$.

![Neuron's model](image1)

**Fig. 1 - Neuron's model**

Upon this neuron model, we define an error model that characterizes the behavior of the neuron and is actually independent of the technological implementation. More specifically, we distinguish:

a. unexpected value of input signal $x_j$: this can be due either to a fault internal to $n_j$ or to faulty interconnections or even to external noise affecting the system;

b. errors affecting the synaptic weight $w_{ij}$ or the associated multiplication by $x_j$: in the absence of any implementation detail, they are to be considered as indistinguishable;

c. errors affecting the summation inside the neuron;

d. errors affecting the non-linear evaluation function; actually, these errors lead to create an unexpected value on $z_i$, and thus are equivalent to error a.

Let us now consider the sensitivity of the neural net to such various errors. Generality leads us to take into account continuous signals, and, in particular, continuous non-linear evaluation functions; the particular case of two-state signals and of step functions can be derived easily.

We analyze first the effects on the individual neuron $n_i$ of errors that modify the input to the evaluation function. As a reference, we consider the two most meaningful instances of evaluation function, namely the sigmoid and the step function.

In the first case, the sigmoid for neuron $n_i$ can be expressed as

$$f_i(s_i) = \frac{1 - e^{-s_i}}{1 + e^{-s_i}}$$  \hspace{1cm} (2)

where $s_i = \sum_j w_{ij} x_j - \theta_i$. Injection of an error $\epsilon$ in $s_i$ produces an error in the output that can be expressed by

$$\epsilon_i(s_i, \epsilon) = f_i(s_i + \epsilon) - f_i(s_i) = 2\frac{e^{-s_i}}{(1 + e^{-s_i})^2} \epsilon$$  \hspace{1cm} (3)

The function is represented by the surface in figure 2, where varying values of the error $\epsilon$ and of the input summation $s_i$ are considered. From this figure we can easily see that errors in input signals, in weight multiplications or in the input summation are equivalent to an error in the computation of the neural evaluation function (i.e. to an error in the neuron's output), from the behavioral point of view. We can therefore refer more concisely to presence of a faulty neuron in the network.

If a step function is taken into account, instead of the sigmoid, the sensitivity of the neuron's output is represented by the surface in figure 3.

![Local error for the sigmoid function](image2)

**Fig. 2 - Local error for the sigmoid function**

![Local error for the step function](image3)

**Fig. 3 - Local error for the step function**

Having examined individual behavior of neurons, it becomes necessary to analyze the response of the whole network to the various types of errors that can be injected in different points. With the exception of errors either affecting one neuron of the last (output) layer or one synapsis leading into it (errors that would not spread to the rest of the net, although they would in principle affect the whole classification), any other error has an influence on the results produced by all neurons in the layers subsequent to the one where the error occurs. We limit ourselves initially to the analysis of effects of a single error; extension to multiple errors will then be outlined. An intuitive, geometrical consideration may be presented first as regards the effects of errors upon the computation; namely, it can be noticed that while errors in the first (input) layer affect the tessellation of the pattern space, by changing the parameters that characterize the hyperplanes without affecting their number, errors in
subsequent layers modify the operations that lead to assembling the various tessels so as to define pattern classes.

In the layered net, assume an error of any of the previous classes affecting neuron \( n_j^i \), i.e., the \( j \)-th neuron of layer \( i \). The error affects its output \( x_j^i \); denote the variation of the output value with respect to the expected one as \( \epsilon \). As a consequence, the summation \( s_j^{i+1} \) computed by any neuron \( n_j^{i+1} \) belonging to layer \( i+1 \) will be affected by an error. We confine ourselves for the moment being to an analysis of effects between the two adjacent layers. The modified value of \( s_j^{i+1} \) is now expressed by:

\[
s_j^{i+1} = \sum_h w_{kh}^{i+1} x_k^i - \theta_h^{i+1} + \sum_h w_{kh}^{i+1} x_k^i + \Delta w_{kh}^{i+1} x_k^i = \sum_h w_{kh}^{i+1} x_k^i + \Delta w_{kh}^{i+1} x_k^i
\]

where \( x_j^i \) is the expected value of the output of \( n_j^i \) in the absence of error, while \( s_j^i \) is the value of the same signal in the presence of an error: i.e., \( s_j^i = x_j^i + \epsilon \). It is then

\[
\hat{s}_j^{i+1} = \hat{s}_j^{i+1} + \Delta s_j^{i+1}
\]

The output of any neuron \( n_j^{i+1} \) in layer \( i+1 \) will in turn be affected by an error

\[
\epsilon_j^{i+1} = \hat{x}_j^{i+1} - x_j^{i+1} = f_{j+1}(s_j^{i+1} + \Delta s_j^{i+1}) - f_{j+1}(s_j^{i+1})
\]

Theoretical analysis allow to state the conditions under which the error \( \epsilon \) introduced into the output \( x_j^i \) of the neuron \( n_j^i \) can be completely masked at the outputs of all neurons of layer \( i+1 \) by properly updating the interconnection weights between neuron \( n_j^i \) and neurons \( n_j^{i+1} \). In other words, we should modify the interconnection weights so as to satisfy the requirement that the error \( \epsilon_j^{i+1} \) at the output of any neuron \( n_j^{i+1} \) of the layer \( i+1 \) be zero for any pattern belonging to the recognized classes. To such purpose, first of all we exclude from the active computation the faulty neuron \( n_j^i \), i.e. we impose that all weights \( w_{kh}^{i+1} \) (connecting the faulty neuron \( n_j^i \) to neurons of the layer \( i+1 \)) be null; we implement thus the “fault confinement” phase that is a pre-requisite for any fault-tolerance policy. Then, we look for the values of \( w_{kh}^{i+1} \) that are able to guarantee error masking. Assume that \( \Delta w_{kh}^{i+1} \) is the new value of the input summation after any weight modification \( \Delta w_{kh}^{i+1} \) between layer \( i \) and layer \( i+1 \), leading to \( \Delta s_j^{i+1} \). It is:

\[
\Delta s_j^{i+1} = \sum_h w_{kh}^{i+1} \Delta x_k^i \]

where consider

\[
\Delta w_{kh}^{i+1} = \begin{cases} -w_{kh}^{i+1}, & \text{for } h = j \text{ and } \forall k \text{ in layer } i+1; \\ w_{kh}^{i+1}, & \text{for } h \neq j \text{ and } \forall k \text{ in layer } i+1, \end{cases}
\]

and \( \Delta w_{kh}^{i+1} \) are the unknown modifications of weights required to achieve error masking. This goal can be achieved thorough weight updating if such technique produces input summation \( s_j^{i+1} \) equal to the corresponding summation \( \hat{s}_j^{i+1} \) in the absence of errors. Thus, it must be:

\[
\Delta s_j^{i+1} = \sum_h w_{kh}^{i+1} \Delta x_k^i = 0
\]

which can be rewritten for each neuron in layer \( i+1 \), by pointing out the unknown quantities, as:

\[
\sum_h \Delta w_{kh}^{i+1} x_k^i = w_{kh}^{i+1} x_k^i
\]

Let now \( H \) be the number of neurons in layer \( i \) and \( K \) the number of neurons in layer \( i+1 \); equation (9) defines a linear system of \( K \) equations in the variables \( \omega_{kh}^{i+1} \). The number of variables is \((H-1) \times K\), since we have \( H \times K \) interconnection weights between layer \( i \) and layer \( i+1 \), but \( K \) weights are fixed by rule (7). The system is linear since the \( \omega_{kh}^{i+1} \) do not depend upon the \( \omega_{kh}^{i+1} \). Besides, for a given pattern presented at the input layer, \( x_k^i \) and \( w_{kh}^{i+1} x_k^i \) may be considered as constant values.

Formally, for a given input pattern, the linear system defined by (9) may be compacted by using a matrix description. Let \( \Omega = [\omega_{kh}^{i+1}] \) (with \( k \in [1,K], h \in [1,H] \) - (j)) be the matrix of unknown weight modifications. We define the row vector \( \mathbf{Y} \) by cascading the rows \( \Omega_k \) of \( \Omega \), i.e. \( \mathbf{Y} = [\Omega_1 \cdots \Omega_k \cdots \Omega_K] \). The column vector \( \mathbf{C} \) is given by the constant vector \( \mathbf{C} = [x_k^i \cdots x_k^i] \).

Finally, the coefficient matrix \( \Phi \) is defined as a block-diagonal matrix, where each non-null block on the main diagonal is equal to the row vector \( \mathbf{X}_i = [x_k^i] \) of the output signal generated by the neurons of layer \( i \) in absence of errors. It has \( K \) rows and \((H-1) \times K\) columns. It can be proved that the rows of \( \Phi \) are linearly independent, since no linear combination of any set of \((K-1)\) rows can generate the remaining row. This can be easily derived from properties of linear algebra by assuming that at least one \( x_k^i \) is null.

From previous definitions, the linear system (9) is thus given by:

\[
\Phi \cdot \mathbf{Y} = \mathbf{C}
\]

where \( \mathbf{Y}^T \) is the transposed of \( \mathbf{Y} \).

Equation (10) has a unique solution for a given input pattern if \( \Phi \) is a square matrix and may be inverted, i.e. \( K = (H-1)K \) (namely, \( H = 2, K > 0 \)) and \( \det(\Phi) \neq 0 \). In such case, the weight modifications that confine the error between the two adjacent layers are given by \( \mathbf{Y}^T = \Phi^{-1} \cdot \mathbf{C} \). Note that, when \( H = 2 \), matrix \( \Phi \) becomes a diagonal matrix and, thus, \( \det(\Phi) \neq 0 \) iff \( x_k^i \neq 0 \), with \( h \neq j \) (i.e. the output signal of the error-free neuron is not null in absence of errors). The probability \( p \) that a solution exists is given by \( p = p(x_k^i \neq 0 \mid h \neq j) \).

If \( \Phi \) is a rectangular matrix, the number of columns is larger than that of rows \((H > 2, K > 0)\), and we may have an infinite number of solutions. In fact, if we can find a square submatrix \( \Psi \) (having rank \( K \)) of \( \Phi \) which is invertible, we have an infinity of solutions of order \((H-2)K\). This means that, for the given input pattern, we can choose arbitrarily the value of \((H-2)K\) weight. Due to the characteristics of the matrix \( \Phi \), it is possible to show that the probability \( p \) that a solution exists is the probability that at least one \( x_k^i \) (with \( h \neq j \)) is not null; i.e. \( p = 1 - \Pi_{x_k^i \neq 0} \), by assuming that \( x_k^i \) are mutually independent.

Extension of (10) to the whole set of \( P \) input pattern classes may be achieved by considering a set of equation (10) for each class \( p \). This leads to equation (11):
where \( \Phi^r = [\Phi_1^r \ | \ \Phi_2^r \ | \ . \ | \ \Phi_p^r], \ \tilde{\Phi}^r = [C_1^r \ | \ C_2^r \ | \ . \ | \ \Phi_p^r], \ C_p^r \), \( \Phi_p \) and \( C_p \) are the matrix \( \Phi \) and \( C \) (defined above) for the class \( p \).

Equation (11) defines a linear system of \( PK \) equations in \((H - 1)K\) variables: its solution (if it exists) gives the weight modifications which guarantee the complete tolerance to the injected error for all input classes. In this system, some rows may be a linear combination of other rows, so that the actual number of equations that should be simultaneously solved will be lower than \( PK \). It can be proved that, if 2 rows are found that are mutually linearly dependent, linear dependency can be found between 2P rows. Should the determinant of the system be null, a possible alternative in order to reach a solution could be to move a small distance the center of each class, thus slightly modifying the parameters of the system without affecting in a relevant way the classification capacity.

If the previous constraints upon \( \Phi \) are not satisfied, no weight modification can be found for the given input patterns to achieve error masking. A further attempt should be made by checking the possibility of obtaining a satisfactory modified weight distribution through all layers from \( i + 1 \) to the output one. An exact solution in this case would anyway be very hard to determine, due to the presence of non-linear functions. We will therefore consider an approximate solution, by first introducing constraints allowing to make the problem manageable.

We assume the effect of the errors allowed to be sufficiently small to keep the operation point in a small interval around its nominal position on the non-linear evaluation function. We then substitute, within such interval, the non-linear function with its linearization, i.e., the tangent in the nominal point. Assuming again an error in the output \( y_i \) of neuron \( n_i \), the error \( \varepsilon \) at the output of any neuron \( n_i \) is now expressed as:

\[
\varepsilon_i^{m+1} = \tilde{x}_i^{m+1} - x_i^{m+1} = f_i^{m+1}(s_i^{m+1} + w_{i,j}^{m+1} \varepsilon) - m_i^{m+1}(s_i^{m+1}) \approx \frac{df_i^{m+1}}{dx_i^{m+1}} \mid_{x_i^{m+1}} w_{i,j}^{m+1} \varepsilon
\]  

The above holds for the individual pattern corresponding to the operation point taken into account; thus, equation (12) must be repeated for all pattern classes. Unless layer \( i + 1 \) is the output one, equation (12) must be extended for all subsequent layers, obtaining for each \( m, m > i + 1 \):

\[
e_i^m = \tilde{e}_i^m - e_i^m = m_i^m(s_i^m + \sum_{k} w_{i,j}^{m-1,m} e_k^m) - m_i^m(s_i^m) \approx \frac{df_i^m}{dx_i^m} \mid_{x_i^m} \sum_{k} w_{i,j}^{m-1,m} e_k^m
\]  

where \( \tilde{e}_i^m \) denote the erroneous values of the outputs produced by each neuron in layer \( m \). A more compact expression of (13), making use of vector notation, is

\[
E_m \approx F_m \ | \ W_{m,m-1} E_{m-1}
\]  

where \( E_m \) is the vector of output errors at layer \( m \), \( S \) is the vector of the summation inputs, \( F \) is the vector of derivatives of \( F \) in \( S \), \( W_{m,m-1} \) is the weight matrix between the adjacent layers. It will be noticed that equation (13) is a recurrent equation, whose solution may be achieved by iterative substitution and involves a relevant computational complexity.

Having thus determined the error propagation through the system, it is necessary to alter the variations of weights throughout all layers (backwards from the output layer up to, and including, layer \( i + 1 \) so as to set an upper bound on the final error produced by the net. To this purpose, we consider two alternative aims; either setting such error to 0 or else minimizing the mean square of the error itself.

Variation of the weights is achieved by considering that each weight \( w_{i,j}^{m-1,m} \) is now modified by \( \Delta w_{i,j}^{m-1,m} \); thus, the input summation to the evaluation function for each neuron in layer \( i + 1 \) is:

\[
\tilde{x}_i^{m} = x_i^{m} + w_{i,j}^{m-1} \varepsilon + \sum_{k} \Delta w_{i,j}^{m-1,k} x_k^m + \Delta w_{i,j}^{m-1} \varepsilon
\]  

The variation of each neuron's output value is as a consequence:

\[
\Delta x_i^m = \tilde{x}_i^m - x_i^m = m_i^m((s_i^m + \Delta w_{i,j}^{m-1} \varepsilon) - m_i^m(s_i^m))
\]  

which (assuming small errors) can be approximated as:

\[
\Delta x_i^m \approx \Delta w_{i,j}^m \varepsilon
\]  

Equation (17) must now be extended to any layer subsequent to \( i + 1 \); in this phase, second-order differences would appear, that we will here neglect, assuming them meaningless by comparison to the other terms. Thus, finally, we evaluate the variation of each neuron's output in any layer \( m, m > i + 1 \), as:

\[
\Delta x_i^m \approx \Delta w_{i,j}^{m-1,m} \varepsilon + \sum_{k} \Delta w_{i,j}^{m-1,k} x_k^{m-1}
\]  

Using vector notation, equation (18) can be written as:

\[
\Delta X_m \approx \Delta W_{m,m-1} \varepsilon
\]  

where \( \Delta X_m \) is the vector of the variations of weights due to the injected errors and to the weight modifications, \( \Delta W_{m,m-1} \) is the matrix of weight modifications between the two adjacent layers, and \( X_{m-1} \) is the vector of outputs generated by neurons belonging to layer \( m - 1 \). Again, equation (18) concerns just one pattern and it should be extended — as in the previous case — to all pattern classes taken into account.

By applying definition (7), we introduce the variables \( \omega \) of the unknown modifications of weights, which can lead to the lowest (possibly null) errors in the outputs of the output layer.

By iterative substitution in the recurrent equation thus obtained, we reach a system of linear equations similar to the following one:

\[
\Delta X_M = C \Delta X_i + \sum_{m=1}^{M} \Delta W_{m,m-1} D_m
\]  

where \( C \) and \( D \) are constant vectors depending on the input classes. The injected error is completely masked at the final outputs of the neural net if \( \Delta X_M = 0 \). This leads to rewrite equation (20) to achieve a linear system similar to (11). Solutions and constraints are thus similar to those discussed above.
for such system. Note, that if no solution exists, we can apply classical dynamic programming techniques to identify a set of values for the variables $w$ that minimize the objective function

$$\phi = \min(||A X_M||).$$

Should the linearization adopted until now be incompatible with the actual variation of operation points (as verified a-posteriori on the network with the modified weight values), equation (20) should have to be treated not as a linear recurrent equation but as a non-linear equation — with the obvious increase in difficulty.

Extension to the case of multiple errors is simple in the assumption that their global effect is still limited enough to allow linearization around the nominal operation point for each fault-free neuron. In this case, effects of the individual errors are simply superimposed to reach the expression of the complete case.

SIMULATIONS AND EXPERIMENTAL RESULTS

The previous analysis of effects of errors and requirements on weight updating to achieve error masking allowed us to state — in theoretical terms — the possibility of reaching fault tolerance (without introducing ad-hoc modifications of the original neural net) through a repeated learning phase, following the instantiation of an error.

To evaluate in quantitative terms the impact of single or multiple errors on the classification capacity of given multi-layered nets we have performed extensive simulations; different net configurations have been analyzed, to take into account the effects of such variables as the number of neurons supporting a given classification scheme, the distribution of neurons over varying numbers of layers, the distribution of neurons among the layers themselves.

Further simulations were then performed to verify, for the same nets, the outcome of a renewed learning phase following the error occurrence; possibility of reaching complete fault tolerance (i.e., of achieving the original classification capacity) or, otherwise, extent of operation error remaining after the learning phase were evaluated.

In the present section, the details of the experiment will be given first, describing the structures of the nets, the learning algorithms and the errors injected; then, the results of the various simulations will be presented and discussed.

The performances of the nets we consider are defined with reference to the capacity of effecting a classification of randomly generated input patterns with respect to a given set of classes. In turn, to grant the greatest generality to the experiment, each class is characterized by a central representative and by a set of related patterns generated by random (white) noise within a given (even reasonably large) range. An example of the tessellation of the space of classified patterns is given in figure 4 for a set of six classes.

The nets capable of classifying such patterns are chosen as multi-layered ones, having at least one — and possibly more — hidden layer. Obviously, all nets have as many neurons in the output layer as the number of classes adopted; otherwise, the different nets are characterized on the basis of the total number of neurons present and of their distribution among layers. Among the different nets, one "as small as possible" is sought, roughly defined as capable of granting correct classification but such that smaller nets could not achieve it (this net has been determined in an experimental way). The evaluation function adopted is a sigmoid for all but the output layers; a step function is adopted for the output layer, to achieve a binary output value. All other output signals have continuous values ranging between -1 and +1.

Initial weights range between -1 and +1: final values may be arbitrary real numbers. The learning algorithm is a classical back-propagation one [4]: different values have been used for the algorithm parameters to check the sensitivity of the evaluations to the characteristics of the algorithm.

Two alternative instances were considered for defining the errors to be injected. The first is an additive error summing to the correct output of a given neuron (or of a set of neurons); the second one consists in a fixed value stuck on the output of a neuron, whatever the ideal value of such output. These two instances, while coherent with the error model defined in the second section, are compatible with fault instances in silicon implementations as well. Simulations have given practically identical results for both error types; as a consequence, we will here present one set of results only, derived from the first alternative.

Refer first to simulation of the net's behavior in the presence of a single error. We considered a) nets with three layers and b) nets with five layers. The number of pattern classes is six for all instances. For case a) we examined the behavior of a large number of nets with varying numbers of neurons (starting from the "lowest" one, as defined above, to a number approximately double of that) and with different distributions of such neurons among the three layers, so as to account for different amounts of synaptic weight memory. Different structures (both with respect to the number and to the distributions of neurons) were examined also for the five-layer nets, although (due to the complexity of the experiment) the analysis was not as exhaustive as in the three-layer case.

A further element of discussion is the position of the faulty neuron within the net; thus, considering for example a three-layer net, it can be seen that while the position of the fault within a layer is not relevant (as it is self-evident), the particular layer in which the fault is positioned is relevant to the value of the final classification error. In figure 5, the percentage error of classification is given for a three-layer net with varying number of neurons in the input and in the hidden layer when the fault is
positioned in the input layer. The surface in fig. 5.a provides the "residue classification error" in the absence of faults, assuming that learning is continued until no improvement is achieved and trying to reach total classification capacity; figure 5.b provides the classification error in the presence of a faulty neuron. The high error ratio when the number of neurons in the first and second layers is small is to be expected, since obviously the corresponding network is near to its "minimum configuration" and no intrinsic redundancy is present. The apparently surprising increase in error for high numbers of neurons can be justified by the sensitivity of stable configurations to highly redundant net structures. In any case, the values obtained for errors are quite high, surprisingly so in view of the general assumption of "intrinsic fault tolerance" for neural nets. In figures 6.a and b, the results of a similar simulation for networks with initial classification error (in the absence of faults) of 96 percent are represented; as it was to be expected the effect of the fault is magnified in this case.

In figure 7 the effect of a faulty neuron in the hidden layer is represented, again with respect to different network structures, while in figure 8 the fault has been positioned in the output layer (these two instances are computed for initial 100 percent classification capacity).
c. increasing numbers of neurons in the input layer, with
the given classification requirements, allow to reach bet-
ter fault-tolerance; this was also to be expected, since the
identical initial tessellation is implemented with a larger
global redundancy of the network.

The effect of repeated learning can now be examined; simu-
lations have been performed for all the previous examples, and
the corresponding surfaces are represented in figures 5.c, 6.c,
7.b, 8.b. It will be noticed that, even after repeated learning,
the residue error in figure 6.c is much higher than in figure 5.c.
This can be interpreted as a lower capacity of error recovery
through repeated learning when the initial learning was not def-
initely perfected (i.e. 100-percent initial classification had not
been reached). Moreover, as it could be intuitively inferred, a
faulty neuron in the output layer is much more critical (even in
the presence of repeated learning) than one in the other layers,
since weight re-distribution has no actual meaning in this case.

Simulations of the same type were then repeated for two
faulty neurons; the corresponding errors are given in figures 9.a
to 12.a, for different locations of the faulty neurons (clearly, it
is useless to repeat the surfaces corresponding to an absence of
faults) while the situation after repeated learning is represented
in figures 9.b to 12.b.

Finally, errors and classification capacity after repetition
of learning have been derived for five-layer networks, always
given the same classes of input patterns. Simulations carried
out allow us to draw meaningful conclusions, namely:
a. results are similar to those obtained for three-layered nets;
b. if the five-layered net has a comparable number of neurons
as the functionally equivalent three-layered one, the input
layer will have lower number of neurons and therefore it will
a rougher tessellation of the pattern class space. As it is to
be expected, therefore, the effect of a fault in such layer is
more relevant than in the three-layer case. Moreover, the
effect of the fault is further magnified by its propagation
through a higher number of layers;

Fig. 8 - Error due to a faulty neuron in the output layer
after fault injection (a), after repeated learning (b)

Fig. 9 - Error due to two faulty neurons in adjacent layers
(one in the input layer and one in the output layer):
after fault injection (a), after repeated learning (b)

Fig. 10 - Error due to two faulty neurons in adjacent layers
(one in the input layer and one in the output layer)
for unperfected learning: after fault injection (a),
after repeated learning (b)
CONCLUDING REMARKS

Theoretical considerations and simulation results on the effect of errors in multi-layered feed-forward neural nets allowed us to reach a number of conclusions, sometimes contrasting with "intuitive" evaluations currently proposed. The first fact is that presence of a faulty element in the network results in an error (even a relevant one) in the classification produced in response to arbitrary input patterns, and that the error is present not only for "minimal" networks, but also for those that are intrinsically redundant (both in terms of neurons and in terms of total synapsis distribution) with respect to minimal classification requirements. As a consequence, it is safe to assert that a neural net per se — i.e., not considering specific implementation forms — does not allow immediate error masking.

A second important fact is that — excepting for the "minimal" nets — a repeated learning phase can lead to a new weight distribution such that afterwards correct classification may still be achieved. Since repeated learning involves a time overhead, there follows that even in the presence of a structure redundancy characterizing the network, fault tolerance can be reached only by adding a measure of time redundancy to the performances of the network (actually, in this context "time redundancy" is not to be meant as it is usual in fault-tolerant systems, since the time increase must be added only to the learning phase and not to each subsequent classification operation).

The next main step to be undertaken will concern, obviously, the mapping of physical faults present in actual implementations upon the behavioral error classes taken into account in the present paper. A point of particular relevance concerns the possible mapping of single physical faults onto multiple behavioral errors, incurred into whenever some kind of time multiplexing is enacted in the implementing architecture (this is the case, typically, of all digital implementations proposed up to now, to overcome the connectivity requirements of neural nets). Such mapping would easily lead to collapse of the implementing architecture in the presence of a few faults. This point, together with the re-learning requirements noted above, lead us to suggest that architectural fault-tolerance solutions — tuned to the particular implementation chosen — ought to be considered even in the case of neural nets implementation, even though the possibility of lowering structure redundancy through exploitation of repeated learning, as well as that of accepting graceful degradation in terms of increasing classification errors, may allow a compromise as far as architecture redundancy is concerned.

Work is at present going on both on the above aspects, linking various implementations with the abstract models considered here, and on the analysis of different neural net structures.

REFERENCES