Modeling of Component Failure in Neural Networks for Robustness Evaluation: An Application to Object Extraction

Ashish Ghosh, Nikhil R. Pal, *Member, IEEE*, and Sankar K. Pal, *Fellow, IEEE*

Abstract—**An investigation on the robustness (or ruggedness) of neural network (NN) based information processing systems with respect to component failure (damaging of nodes/links) is done. The damaging/component failure process has been modeled as a Poisson process. To choose the instants or moments of damaging, statistical sampling technique is used. The nodes/links to be damaged are determined randomly.**

As an illustration, the model is implemented and tested on different object extraction algorithms employing Hopfield's associative memory model, Gibbs random fields, and a self-organizing multi-layer neural network. The performance (hence robustness of the underlying network model) of these algorithms is evaluated in terms of percentage of pixels correctly classified under different noisy environments and different degrees and sequences of damaging. The deterioration in the output is seen to be very small even when a large number of nodes/links are damaged.

I. Introduction

A neural network (NN) [1]–[7] based system consists of a large number of neurons with massive connectivity among them. Local connectivity among the neurons/nodes (computing elements) being very high and the storage of information being distributed, the approach is claimed to be highly robust and can be applied even when information is ill-defined and/or defective/partial or noisy. If some of the components fail to work, the other components adjust themselves (during iterative learning) in such a manner that the output is not deteriorated much. As an NN-based system consists of a large number of components, the possibility of some of its components (nodes or links) to fail to work is very high. Here lies the necessity of investigating the performance of an NN–based system under failure of some of its components.

In the present work an attempt is made to provide a model for studying the robustness or ruggedness under component failure of NN-based systems empirically. This is done by damaging some of the nodes and/or links of a system and studying the change in its performance. The failure process is viewed as a pure death process. For our investigation we made a set of assumptions about the failure process. Incidentally, these assumptions led the failure process follow Poisson model. In this context we mention that Poisson distribution is often used to model different failure processes; but for all failure processes it may not be applicable. Under this model, the nodes and/or links to be damaged are chosen randomly. The time instants (i.e., when a damage occurs) are determined by drawing random samples from the appropriate probability distribution [8], [9].

Though the model is valid, in general, for any NN-based information processing system (supervised and unsupervised), the problem of image segmentation and object extraction (unsupervised system) is considered here for an extensive demonstration of the validity of the model. Three different algorithms [10]–[12] which extract compact object regions from noisy environments using Hopfield's associative memory, Gibbs random field, and a self-organizing multi-layer NN are chosen for this purpose.

To demonstrate the utility of the proposed failure model, performance of these algorithms is evaluated in terms of percentage of pixels correctly classified. Outputs are evaluated in four different situations: i) no damage, ii) damaging only nodes, iii) damaging only links, and iv) damaging both nodes and links simultaneously. The simulation study is done on various noisy versions of a synthetic image using different amounts and sequences of damaging. Note that the measurement of robustness is problem dependent. For example, if an NN is used in an optimization problem, its robustness may be evaluated computing the difference between the values of the objective function with and without component failure.

The rest of the article is organized as follows. In Section II, the theories of modeling and sampling of component failures in neural networks are described. Section III involves application of the theory to a network. Section IV provides the implementation methodology and simulation results. Concluding remarks are given in Section V.

II. Modeling and Sampling of Failures in a Neural Network

A. Modeling of Failures

Let us consider a neural network system with N components, where a component could be a node (processor) or a link. During the operation of the network some of its components may fail. We make the following assumptions about the failure process:

Manuscript received June 24, 1992; revised March 10, 1993, August 23, 1993, and March 11, 1994.

The authors are with the Machine Intelligence Unit, Indian Statistical Institute, Calcutta 700 035, India.

IEEE Log Number 9404857.

i) The system has N identical components at the time instant $t = 0$ (when the operation of the network starts).
ii) If a component fails, it fails forever (no repair or replacement).
iii) Failure of components occurs at the average rate of μ per unit time.
iv) The probability of an event occurring between time t and $t + h$ depends only on the length of h, i.e., the probability does not depend on either the number of events that has occurred up to time t or the specific value of t, i.e., the probability density function has stationary increments.
v) The probability of a failure during a very small interval of time h is positive but less than one, i.e., not certain.
vi) At most one failure can occur during a very small interval of time h.

Let $p_n(t)$ be the probability that the system has n components active at time instant t, i.e., $N - n$ failures during time interval [0,t]. It can be shown [8] that under the assumptions i–vi, $p_n(t)$ is given by the formula

$$p_n(t) = \frac{e^{-\mu t}(\mu t)^{N-n}}{(N-n)!}, \quad n = 1, 2, \cdots, N \tag{1}$$

and

$$p_0(t) = 1 - \sum_{n=1}^{N} p_n(t). \tag{2}$$

Thus we see that $p_n(t)$ is a truncated Poisson distribution with mean μt.

If $f(t)$ is the probability density function (pdf) of the inter-failure time (i.e., time interval between two successive failures), then it can be shown that for the earlier Poisson failure process, $f(t)$ is given by

$$\begin{aligned} f(t) &= \mu e^{-\mu t} \quad t > 0 \\ &= 0 \qquad\quad t \le 0. \end{aligned} \tag{3}$$

Thus, when the failure process is governed by a Poisson distribution, the inter-failure time is then described by an exponential distribution (3) with expected value (mean)

$$\begin{aligned} E(t) &= \int_0^\infty t f(t) dt \\ &= \frac{1}{\mu}. \end{aligned} \tag{4}$$

B. Sampling

To simulate the failure process, one needs to draw random samples from the exponential distribution (3). Before describing the exact algorithm, let us first consider the general strategy for sampling from any distribution.

Let $f(x)$ be the pdf of the random deviate x, and $F(x)$ be the cumulative density function (cdf) of x, i.e.,

$$F(x) = \int_{-\infty}^{x} f(t) dt. \tag{5}$$

It can be easily shown that the random variable $y = F(x)$ is uniformly distributed over [0, 1], regardless of the distribution of x. Hence, if R is a random number drawn from uniform [0, 1], then $x = F^{-1}(R)$ is a random sample from the pdf $f(x)$. Therefore, sampling from any distribution can be done using the following simple method having two steps.

Step 1: Generate a random number R in [0, 1] and assign it to $F(x)$.

Step 2: Solve for x from $R = F(x)$.

The above sampling method is known as method of inversion.

Sampling from Exponential Distribution: For an exponential distribution the pdf is

$$\begin{aligned} f(t) &= \mu e^{-\mu t} \quad \mu > 0, \qquad t > 0 \\ &= 0 \qquad\qquad\qquad\quad t \le 0. \end{aligned} \tag{6}$$

Then

$$F(t) = \int_0^t \mu e^{-\mu x} dx = 1 - e^{-\mu t}.$$

If the random number drawn is R then

$$R = F(t)$$

or

$$R = 1 - e^{-\mu t}$$

or

$$t = -\frac{1}{\mu} \ln(1 - R) = -\frac{1}{\mu} \ln R. \tag{7}$$

The last step is possible because if R is a random number on [0, 1] then so is $(1 - R)$ and we can replace $(1 - R)$ by R for convenience.

It has been established before that if the failure process is described by a Poisson distribution, then the time between the occurrence of failures (interfailure time) must follow the corresponding exponential distribution. Thus to simulate the component failure process described by the Poisson distribution with mean μt, over a time period [0,T], all one has to do is to sample the corresponding exponential distribution with mean $1/\mu$ as many times as necessary until the sum of the corresponding exponential random samples generated exceeds T for the first time. It can further be explained as follows.

Suppose R_i is the ith random sample drawn from uniform [0,1], then

$$t_i = -\frac{1}{\mu} \ln R_i \tag{8}$$

is the ith sample from the exponential distribution (3). Therefore

$$T_i = \sum_{j=1}^{i} t_j \tag{9}$$

gives the time instant when the ith (component) failure occurs. The process is repeated for the maximum number of times (K, say) such that $T_k \le T$.

III. Application of Failure Model to Robustness Evaluation of Neural Networks

In any system components may fail with passage of time. In the case of NN-based systems the components are the neurons/nodes and links. So in such systems, some of the neurons or links or both may get damaged over time. NN-based information processing systems are normally claimed to

be robust under components failure as the NN architectures involve massive processing elements and connectivity among them (mostly with a few redundant components). The systems are also supposed to be noise insensitive and suitable for ill-defined/partial information.

The robustness of a neural network system can be investigated under three different situations:

- failure of only nodes,
- failure of only links, and
- failure of both nodes and links.

We have already seen in Section II-A that if the components are identical and failures are independent, then the interfailure time follows an exponential distribution. Therefore, while considering failures of only nodes or only links, we can use exponential distribution (3) to draw samples for interfailure time. Section II-B discusses how to compute the time instant T_i of the ith failure, but does not mention about the selection of the node (link) to damage at that instant T_i. Since all nodes (or all links) are assumed to be identical, one can select a node (to be damaged) randomly in such a manner that each node (link) has an equal chance of being selected.

The situation becomes a little more complicated when both nodes and links are allowed to fail. A node failure effectively disables several links connected to it. On the other hand, failure of a set of links may effectively disable a node. Some of the assumptions stated in Section II-A are not valid, if we consider both node and link failures together, because the set of components are not identical and failure of one component may cause effective failure of another. We propose to investigate the system under the following two simplifying situations.

Case I: Do not differentiate between nodes and links, i.e., treat either of them as a component. The damaging process is then viewed to follow a single Poisson distribution.

Case II: Nodes and links are treated as two different components. The damaging process will then be governed by two different Poisson distributions, one for the nodes and the other for the links. No interdependence between the two distributions is assumed.

The components (nodes/links) which are getting damaged can be chosen randomly. When we are damaging only one type of component (i.e., either node or link), the easiest way to select a component is to assign a unique number to each component and use random numbers in such a way that each node has an equal chance of inclusion. If an already selected component reappears, then one has to ignore that drawing. When one is damaging both nodes and links together the selection process will be as follows depending on the previously mentioned two cases. Suppose there are M nodes and N links.

Case I: Each component is assigned a unique integer number in the range [1, $M+N$], and a component can be selected by drawing randomly an integer in that range.

Case II: Assign each node a unique integer in the range [1, M] and each link a unique integer in the range [1, N]. Selection of a node can be done by drawing a random integer in $[1, M]$. Similarly links are selected.

Note that Case I does not discriminate between nodes and links and hence interdamage times are drawn from one distribution for both the nodes and links. Case II, on the other hand, does discriminate between nodes and links and uses two independent distribution (one for links and the other for nodes).

The model of damaging process developed in this section is, in general, applicable for any NN-based system (both supervised and unsupervised). In the following sections an application of the model is demonstrated on some NN-based object extraction algorithms. For a supervised system, this failure can occur in either of the learning and testing phases. For an unsupervised system there is no testing phase, and the failure can occur during updation of weights/status for unsupervised learning.

IV. Application to Object Extraction Problems

Let us consider three object extraction algorithms [10]–[12] for demonstration of the validity of the model. The first two algorithms use modified version of Hopfield's network model, whereas the third one deals with a self-organizing multilayer neural network. Let us first brief the algorithms for the convenience of the readers before describing the present experimental procedure and results of investigation.

A. Algorithms

Algorithm 1 [10]: In this algorithm the task of image segmentation is formulated as an optimization problem and solved by a modified version of Hopfield's model. It involves minimization of an objective function

$$E = -\sum_i \sum_j W_{ij} V_i V_j - \sum_i V_i I_i \tag{10}$$

where W_{ij} is the connection strength between the ith and the jth neurons, I_i ($\in [0,1]$) is the input bias to the ith neuron and V_i ($\in [0,1]$) is the output status of the ith neuron.

In the present study we have chosen a 3×3 neighborhood. Here, since the number of neighbors is fixed (8), the input value to a neuron lies in the domain $[-8, 8]$. A polynomial function

$$\begin{aligned} g(x) &= -1 && \text{if } x \le -8 \\ &= \tfrac{1}{2^{3n}}(x+8)^n - 1 && \text{if } -8 \le x \le 0 \\ &= 1 - \tfrac{1}{2^{3n}}(8-x)^n && \text{if } 0 \le x \le 8 \\ &= 1 && \text{if } x \ge 8 \end{aligned} \tag{11}$$

defined over the finite domain $[-8, 8]$ is used as an input/output transfer function.

The weights and input biases are given in such a way that the network self-organizes to form compact clusters. The dynamics is similar to that of Hopfield's model.

Algorithm 2 [11]: A modified version (both architecture wise and neuron characteristic wise) of Hopfield's model is used to determine the maximum aposterior probability (MAP) estimate of a scene from a noise corrupted realization (modeled by Gibbs random field). Here also, a single neuron is assigned to every pixel, and it is connected only to all of its nearest neighbors. The energy function of the network was designed

to be

$$E = -\sum_i \sum_j W_{ij} V_i V_j - \frac{1}{\sigma^2} \sum_i V_i I_i + \frac{1}{2\sigma^2} \sum_i V_i^2 \quad (12)$$

such that its minimum value corresponds to the MAP estimate of the scene.

Here σ is the noise level and the other symbols have the same meaning as in Algorithm 1.

The operations (dynamics) are similar to that in Algorithm 1. In the stable state the set of neurons with ON status constitute the object region.

Algorithm 3 [12]: This algorithm is based on the integration of the concept of fuzzy sets and neural network models by incorporating various fuzziness measures in a multi-layer (one input, one hidden, and one output layer in the present investigation) network for object extraction. The architecture is a feedforward one with back propagation of error and is used as an unsupervised classifier. Each neuron is connected to the corresponding neuron in the previous layer and to its neighbors. There also exists a feedback path from the output to the input layer. The status of neurons in the output layer is described as a fuzzy set. A fuzziness measure [13] of this set is used as a measure of error of the system (instability of the network).

The input value (U_i) to the ith neuron in any layer [except the input layer] is

$$U_i = \sum_j W_{ij} V_j \quad (13)$$

where W_{ij} is the connection strength between the ith neuron of one layer and jth neuron of the previous layer. j can either belong to the neighborhood of i or $j = i$ of the previous layer. A sigmoidal transfer function is then applied to get the output status of the neurons in this layer. These outputs are then fed as input to the next layer. Starting from the input layer, this way the input pattern is passed on to the output layer and the corresponding output states are calculated. The output value of each neuron lies in [0, 1].

After the weights have been adjusted properly, the output of the neurons in the output layer is fed back to the corresponding neurons in the input layer for the next pass. The iteration (updating of weights) continues until the network stabilizes, i.e., the error value (measure of fuzziness) becomes negligible. When the network stabilizes the output status of the neurons in the output layer becomes either zero or one. The neurons having output value zero constitute one group and those having output value one constitute the other group.

In the present investigation, the quadratic index of fuzziness [13] is used as the error measure. Accordingly, the weight updating rule becomes

$$\triangle W_{ji} = \begin{cases} \eta(-V_j) f'(I_j) V_i & \text{if } 0 \le V_j \le 0.5 \\ \eta(1 - V_j) f'(I_j) V_i & \text{if } 0.5 < V_j \le 1.0 \end{cases} \quad (14)$$

for the output layer and

$$\triangle W_{ji} = \eta \left(\sum_k \delta_k W_{kj} \right) f'(I_j) V_i \quad (15)$$

for hidden layers, where $\delta_k = -\frac{\partial E}{\partial I_k}$ and η is a proportionality constant.

B. Method of Implementation and Experimental Results

To demonstrate the utility of the proposed failure model described in Section II, a computer simulation study has been done on the three-object extraction algorithms described in Section IV-A. The only parameter for Poisson/exponential distribution, namely the mean rate μ, is estimated as follows. (Note that since these object extraction algorithms are unsupervised, failures occur only during updation of weights and/or output status of the neurons.)

Implementation Details: Suppose a single-layer network has C components and requires T seconds for I iterations on a monoprocess system. Then for parallel implementation, time/iteration $= \frac{T}{CI}$ and the total processing time is TP $= \frac{T}{C}$. Now if we want to damage a total of D components then an estimate of μ is

$$\mu = \frac{D}{\text{TP}}.$$

Let the inter-damage time periods (samples) be $t_j, j = 1, 2, \cdots, L (L \approx D)$. Now for each of the L time instants select a component to be damaged. In other words, select L components and damage the ith selected component after T_i seconds, where

$$T_i = \sum_{j=1}^{i} t_j.$$

Now if the ith component is to be damaged at a time T_i, and T_i falls in kth iteration, then for the kth and subsequent iterations assume the ith component as damaged. Since the instants or moments of damaging are chosen randomly, both the iteration number and the number of iterations that are taking part in the damaging process vary from simulation to simulation.

For a multilayer (m-layer, say) neural network, let there be C components per layer, and it requires T seconds for stabilization. (Bypassing it may be mentioned that if the multilayer neural network implements a supervised system, the stabilization time T includes both training phase and testing phase.) The ith $(i > 1)$ layer can work only after $(i - 1)$th layer has finished its task. Since there exists a feedback path from the output to the input layer, the output layer may be considered to be the predecessor of the input layer, i.e., the input layer can work (except the starting moment) only after the output layer has completed its operation. Thus parallel implementation can be done only in the C components in a layer. Between layer operations are always sequential. Hence, the total time required for parallel implementation of such a system is TP $= \frac{T}{C}$. Average stabilization time T for parallel implementation on VAX-8650 computer of Algorithms 1, 2, and 3 is found to be 45.78 milli seconds (ms), 45.78 ms, and 22.89 ms, respectively, in the present experiment.

The performances evaluation criterion used in the present investigation is described in the next section.

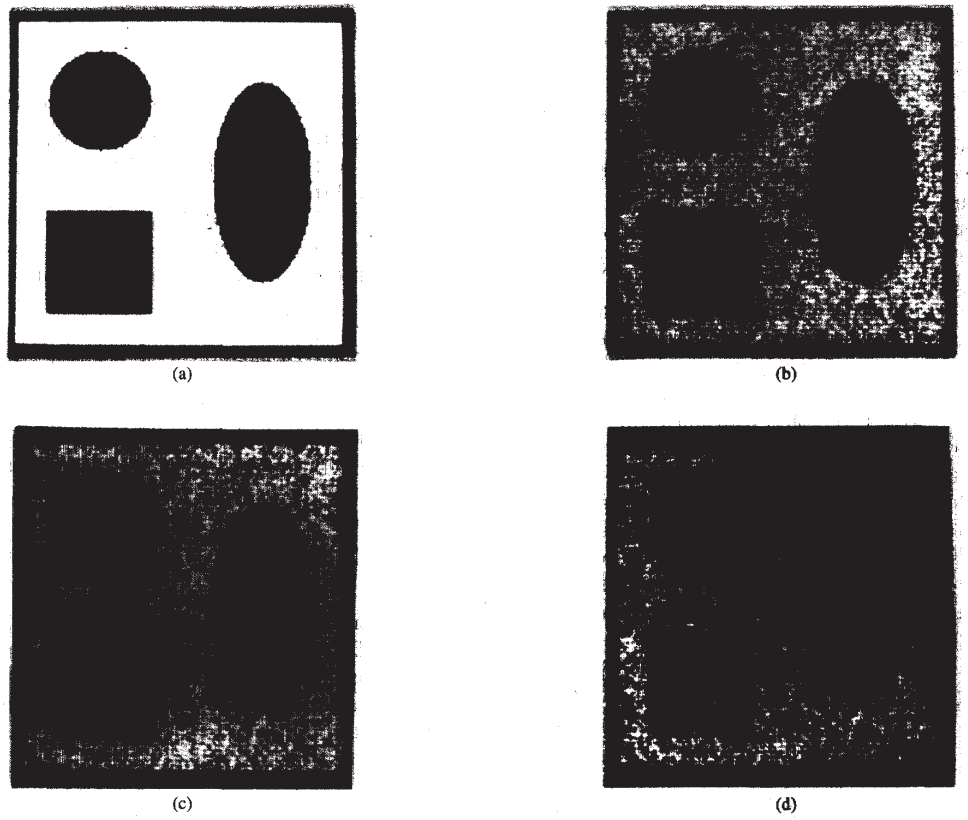

Fig. 1. Input images (a) Original image. (b) Noisy image with σ =10. (c) Noisy image with σ =20. (d) Noisy image with σ =32.

C. Performance Evaluation

By performance of an algorithm we mean the quality of the output it produces. The quality of the outputs can be measured either by visual inspection or by quantitative evaluation. Quantitative evaluation is possible (correctly) only if the actual output (target) is known. Percentage of pixels correctly classified is considered to be a quantitative index of performance. Let N_1 be the number of pixels that actually belong to object, but are classified as background pixels by an algorithm, and N_2 be the number of background pixels that are labeled as object pixels; then the percent of pixels correctly classified (pcc) by the algorithm is given as

$$\text{pcc} = \left(1 - \frac{N_1 + N_2}{M \times N}\right) \times 100. \tag{16}$$

Results and Discussions: The present simulation study is done for all the three different object extraction algorithms (described in Section IV-A) using three different noisy versions (e.g., σ =10, 20, 32) of a synthetic image of size 128 × 128 (Fig. 1). So there are 128 × 128 nodes in each layer of the network architectures used for this purpose.

Through a series of experiments (see [10]–[12]) it was found that if for a node, the number of neighbors is four (second-order connectivity), the output becomes more noise sensitive than that when eight neighbors (third-order neighborhood) are used. For higher order (>3) connectivity, improvement in performance is seen to be negligible compared to that of the third order connectivity. In other words, increasing connectivity after third-order increases the redundancy in links. More over, since it is extremely difficult to find out the optimal architecture for such a neighborhood information-based system (e.g., one needs to test the utility, for a certain goal of the net, of each node and link for all possible combinations with other nodes and links), we considered, for the sake of simplicity, a third-order connectivity (symmetric eight neighbor connection). Note that the network may have some redundant links, although it does not have any redundant node.

The original image is corrupted by adding $N \sim (0, \sigma^2)$ i.i.d. noise. The experiment is done under the following four conditions:

1) without damaging nodes or links,
2) damaging nodes only,
3) damaging links only, and
4) damaging nodes and links both.

Investigation has been done with 2%, 5%, and 10% of component failures for different sequences of damaging. Again, in each case 10 simulations (corresponding to different sequences of damaging) were performed. The dispersion of the obtained pcc values (collected after 50 iterations) was found to be much less. The average percentages of pixels correctly

TABLE I
CLASSIFICATION ACCURACY AFTER NODE DAMAGE

Input image with noise level (σ)	Number of nodes damaged			
	0 (0%)	328 ($\simeq$ 2%) (μ = 7.16)	819 ($\simeq$ 5)% (μ = 17.90)	1638 ($\simeq$ 10%) (μ = 35.80)
10	99.71	99.67	99.68	99.40
20	99.12	99.11	99.04	98.83
32	97.55	97.44	97.54	97.18

TABLE II
CLASSIFICATION ACCURACY AFTER LINK DAMAGE

Input image with noise level (σ)	Number of links damaged			
	0 (0%)	2622 ($\simeq$ 2%) (μ = 57.30)	6554 ($\simeq$ 5)% (μ = 143.16)	13107 ($\simeq$ 10%) (μ = 286.32)
10	99.71	99.66	99.55	99.48
20	99.12	99.11	99.06	98.77
32	97.55	97.57	97.70	97.61

TABLE III
CLASSIFICATION ACCURACY AFTER NODE-LINK DAMAGE (NODES AND LINKS ARE TREATED AS SEPARATE COMPONENTS)

Input image with noise level (σ)	Number of nodes+links damaged			
	0 (0%)	328+2622 ($\simeq$ 2%) (μ_n = 7.16) (μ_l = 57.30)	819+6554 ($\simeq$ 5)% (μ_n = 17.90) (μ_l = 143.16)	1638+13107 ($\simeq$ 10%) (μ_n = 35.80) (μ_l = 286.32)
10	99.71	99.54	98.92	99.05
20	99.12	99.02	98.92	98.38
32	97.55	97.67	97.60	96.88

TABLE IV
CLASSIFICATION ACCURACY AFTER NODE-LINK DAMAGE (NODES AND LINKS ARE TREATED AS A SINGLE COMPONENT)

Input image with noise level (σ)	Number of (nodes+links) damaged			
	0 (0%)	2952 ($\simeq$ 2%) (μ = 64.44)	7371 ($\simeq$ 5)% (μ = 161.10)	14742 ($\simeq$ 10%) (μ = 322.20)
10	99.71	99.69	99.43	99.04
20	99.12	98.63	98.86	98.67
32	97.55	97.35	97.38	96.95

TABLE V
CLASSIFICATION ACCURACY AFTER NODE DAMAGE

Input image with noise level (σ)	Number of nodes damaged			
	0 (0%)	328 ($\simeq$ 2%) (μ = 7.16)	819 ($\simeq$ 5)% (μ = 17.90)	1638 ($\simeq$ 10%) (μ = 35.80)
10	99.38	99.26	98.93	98.49
20	98.80	98.57	98.40	97.97
32	98.03	97.95	97.85	97.28

TABLE VI
CLASSIFICATION ACCURACY AFTER LINK DAMAGE

Input image with noise level (σ)	Number of links damaged			
	0 (0%)	2622 ($\simeq$ 2%) (μ = 57.30)	6554 ($\simeq$ 5)% (μ = 143.16)	13107 ($\simeq$ 10%) (μ = 286.32)
10	99.38	99.22	98.93	98.55
20	98.80	98.75	98.62	98.30
32	98.03	97.90	97.71	97.72

TABLE VII
CLASSIFICATION ACCURACY AFTER NODE-LINK DAMAGE (NODES AND LINKS ARE TREATED AS SEPARATE COMPONENTS)

Input image with noise level (σ)	Number of nodes+links damaged			
	0 (0%)	328+2622 ($\simeq$ 2%) (μ_n = 7.16) (μ_l = 57.30)	819+6554 ($\simeq$ 5)% (μ_n = 17.90) (μ_l = 143.16)	1638+13107 ($\simeq$ 10%) (μ_n = 35.80) (μ_l = 286.32)
10	99.38	98.97	98.50	97.97
20	98.80	98.68	98.59	97.75
32	98.03	97.72	97.53	97.30

TABLE VIII
CLASSIFICATION ACCURACY AFTER NODE-LINK DAMAGE (NODES AND LINKS ARE TREATED AS A SINGLE COMPONENT)

Input image with noise level (σ)	Number of (nodes+links) damaged			
	0 (0%)	2952 ($\simeq$ 2%) (μ = 64.44)	7371 ($\simeq$ 5)% (μ = 161.10)	14742 ($\simeq$ 10%) (μ = 322.20)
10	99.38	99.05	98.46	97.95
20	98.80	98.63	98.36	97.97
32	98.03	97.64	97.02	96.60

TABLE IX
CLASSIFICATION ACCURACY AFTER NODE DAMAGE

Input image with noise level (σ)	Number of nodes damaged			
	0 (0%)	656 ($\simeq$ 2%) (μ = 28.63)	1638 ($\simeq$ 5)% (μ = 71.58)	3276 ($\simeq$ 10%) (μ = 143.16)
10	99.58	99.58	99.57	99.42
20	98.90	98.90	98.88	98.74
32	97.17	97.18	97.37	97.24

TABLE X
CLASSIFICATION ACCURACY AFTER LINK DAMAGE

Input image with noise level (σ)	Number of links damaged			
	0 (0%)	5898 ($\simeq$ 2%) (μ = 257.70)	14746 ($\simeq$ 5)% (μ = 644.25)	29491 ($\simeq$ 10%) (μ = 1288.49)
10	99.58	99.58	99.58	99.51
20	98.90	98.90	98.89	98.90
32	97.17	97.18	97.23	97.24

classified under all the conditions mentioned earlier are given in Tables I–XII. Tables I–IV correspond to Algorithm 1, while Tables VI–VIII correspond to Algorithm 2 and Tables IX–XII ccorrespond to Algorithm 3. In Tables III, VII, and XI, μ_n and μ_l denote the mean rate (μ) for nodes and links, respectively.

From the tables it is evident that when either only nodes or only links are damaged (even a high percentage $\simeq$ 10), the performance is not deteriorated much for all the algorithms. But, if both the nodes and links (Tables III–IV, VII–VIII, and XI–XII) are damaged, performance is degraded slightly more. Even under this condition the results are better for

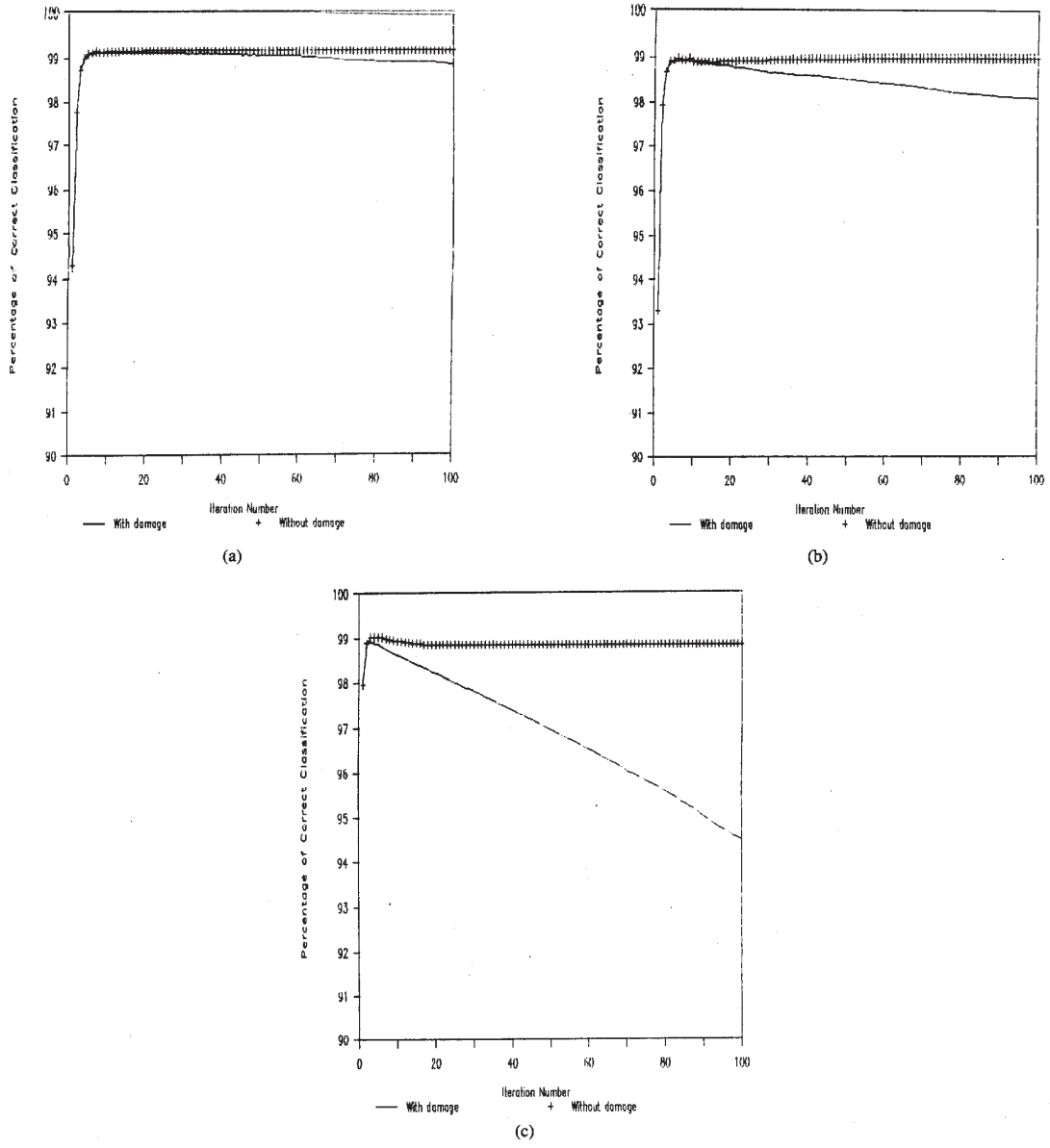

Fig. 2. Variation of percentage of correct classification with iteration (a) for Algorithm 1, (b) for Algorithm 2, and (c) for Algorithm 3.

TABLE XI
CLASSIFICATION ACCURACY AFTER NODE-LINK DAMAGE (NODES AND LINKS ARE TREATED AS SEPARATE COMPONENTS)

Input image with noise level (σ)	Number of nodes+links damaged			
	0 (0%)	656+5898 ($\simeq$ 2%) (μ_n = 28.63) (μ_l = 257.70)	1638+14746 ($\simeq$ 5)% (μ_n = 71.58) (μ_l = 644.25)	3276+29491 ($\simeq$ 10%) (μ_n = 143.16) (μ_l = 1288.49)
10	99.58	99.57	99.51	97.88
20	98.90	98.89	98.82	97.05
32	97.17	97.20	97.08	95.17

TABLE XII
CLASSIFICATION ACCURACY AFTER NODE-LINK DAMAGE (NODES AND LINKS ARE TREATED AS A SINGLE COMPONENT)

Input image with noise level (σ)	Number of (nodes+links) damaged			
	0 (0%)	6554 ($\simeq$ 2%) (μ = 286.30)	16384 ($\simeq$ 5)% (μ = 715.80)	32767 ($\simeq$ 10%) (μ = 1431.60)
10	99.58	99.58	99.51	98.38
20	98.90	98.89	98.90	97.69
32	97.17	97.18	97.08	95.17

the first two algorithms compared to the third one. For the third algorithm the outputs are, to some extent, dependent on the initial gray values of the input image (since the error is calculated by considering the closest approximation as the target output, thereby initially mislabeling some pixels). Normally this initial misclassification gets corrected later on by incorporating neighboring nodes' information while updating links. But when a large number of nodes and links are damaged some of the initial mislabeling fail to be corrected, thereby deteriorating the performance. This, possibly, is the reason of deterioration in the results of the Algorithm 3. Since the performance is not hampered that much even when a large number of nodes and/or links are damaged, the algorithms can be said to be robust to a great extent.

It may be mentioned here that for links, it is not possible to make comments on whether the architecture, even after cutting some of the links, had redundancy or not. The deterioration of the output only indicates that some of the nonredundant (useful) links are lost. Whether a link or a node is redundant or not at some instant of time depends on the organization (presence of active nodes and links) of the net at that instant.

Note that the percentage of pixels correctly classified can accurately be calculated only after the network is converged (i.e., the status of neurons is either -1 and one, or zero and one). While the network is in operation the output status is not binary. To compute the pcc value in such an intermediate stage, one can choose the middle most value of the range of continuous output as the threshold and assign the status as -1 (or zero) for those neurons having output less than the threshold and as one for those neurons having output greater than the threshold. Fig. 2 shows, as an illustration, the variation of pcc (averaged over 10 different simulations as in the tables) computed using the above scheme with iteration when the image in Fig. 1(c) is used as input. Here 5% damage of links and nodes was done treating them as separate components. As expected, pcc value decreases with increase of component failure, whereas it is constant when there is no damage.

V. Conclusion

Robustness with respect to component failure is one of the most important characteristics of NN-based information processing systems. No result has been suggested (as far as the knowledge of the authors go) to date in support of this claim. Present work investigated this feature. This is performed by damaging some of the nodes and/or links and studying the performance of the system. The damaging process is viewed as a pure death process. A set of assumptions is made about the failure process which, incidentally, led the failure process follow Poisson model.

To demonstrate the utility of the model developed, we considered three different image segmentation algorithms which employ Hopfield's net, Gibbs random field, and a self-organizing multi-layer neural network for extracting compact object regions from noisy environments. Robustness of these NN-based algorithms is evaluated under different amount of noises and damaging. Performance is seen to be satisfactory even when a high percentage of nodes/links are damaged. This, in turn, establishes the utility of the robustness evaluation model in addition to examining the performances of some segmentation algorithms.

We mention here that the convergence time of electronic implementation of Hopfield's network is of the order of milliseconds. Therefore, not many components can fail during a single run of the net. In the present investigation also, the convergence time for parallel implementation on a digital platform is also of the order of milli seconds. The percentage of failure of components is considered here to be very high just to investigate the performance of the systems even under such a severe damage. Since the failure process is random, it is difficult to predict at what time a component will fail. Thus even if a single component fails during the operation of a network for which the convergence time is of the order of milli seconds, we can check the robustness of the system with respect to that component.

Acknowledgment

The authors would like to thank the reviewers for their stimulating comments.

References

[1] J. J. Hopfield, "Neurons with graded response have collective computational properties like those of two state neurons," *Proc. Nat. Academy Sci.*, pp. 3088–3092, 1984.
[2] D. E. Rumelhart *et al.*, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, vol. 1.* Cambridge, MA: MIT Press, 1986.
[3] S. Grossberg, Ed., *Neural Networks and Natural Intelligence.* Reading, MA: MIT Press, 1988.
[4] Y. H. Pao, *Adaptive Pattern Recognition and Neural Networks.* New York: Addison-Wesley, 1989.
[5] T. Kohonen, *Self-Organization and Associative Memory.* Berlin: Springer-Verlag, 1989.
[6] P. D. Wassermann, *Neural Computing: Theory and Practice.* New York: Van Nostrand Reinhold, 1990.
[7] K. Fukushima, "Neocognitron: A self-organizing multilayer neural network model for a mechanism of pattern recognition unaffected by shift in position," *Biological Cybern.*, vol. 36, pp. 193–202, 1980.
[8] H. A. Taha, *Operations Research: An Introduction.* New York: Macmilan, 1982.
[9] A. Gupta, *Groundwork of Mathematical Probability and Statistics.* New Delhi, India: Academic, 1983.
[10] A. Ghosh, N. R. Pal, and S. K. Pal, "Object background classification using Hopfield type neural network," *Int. J. Pattern Recog. Artif. Intell.*, vol. 6, no. 5, pp. 989–1008, 1992.
[11] ——, "Image segmentation using a neural network," *Biol. Cybern.*, vol. 66, no. 2, pp. 151–158, 1991.
[12] ——, "Self-organization for object extraction using multilayer neural network and fuzziness measures," *IEEE Trans. Fuzzy Syst.*, vol. 1, no. 1, pp. 54–68, 1993.
[13] S. K. Pal and D. D. Majumder, *Fuzzy Mathematical Approach to Pattern Recognition.* New York: Wiley, 1986.

Ashish Ghosh received the B.E. degree in electronics and telecommunication from the Jadavpur University, Calcutta, in 1987, and the M.Tech. and Ph.D. degrees in computer science from the Indian Statistical Institute, Calcutta, in 1989 and 1993, respectively.

At present he is a Computer Engineer in the Machine Intelligence Unit of the Indian Statistical Institute, Calcutta. His research interests include neural networks, genetic algorithms, image processing, fuzzy sets and systems, and pattern recognition.

Dr. Ghosh received the Young Scientist Award in computer science from the Indian Science Congress Association in 1992.

Nikhil R. Pal (M'91) received the B.Sc. degree with honors in physics and M.B.A. from the University of Calcutta in 1979 and 1982, respectively. He received M.Tech. (Comp. Sc.) and Ph.D. (Comp. Sc.) degrees from the Indian Statistical Institute in 1984 and 1991, respectively.

Currently he is with the Machine Intelligence Unit of Indian Statistical Institute, Calcutta. From 1991–1993 he was with the Computer Science Department of the University of West Florida. He was a guest faculty of the University of Calcutta also. His research interest includes image processing, fuzzy sets theory, measures of uncertainty, neural networks, and genetic algorithms.

Dr. Pal is an Associate Editor of the *International Journal of Approximate Reasoning* and IEEE TRANSACTIONS ON FUZZY SYSTEMS.

Sankar K. Pal (M'81–SM'84–F'92) received the B.Sc. (Hons.) degree in physics and the B.Tech., M.Tech., and Ph.D. degrees in radiophysics and electronics in 1969, 1972, 1974 and 1979, respectively, from the University of Calcutta, India. In 1982 he received another Ph.D. degree in electrical engineering along with the DIC degree from Imperial College, University of London.

He is a Professor and Founding Head of the Machine Intelligence Unit at the Indian Statistical Institute, Calcutta. He served as a Professor-in-Charge of the Physical and Earth Sciences Division, Indian Statistical Institute during 1988–90. He was also a Guest Lecturer (1983–86) in Computer Science, Calcutta University. His research interests mainly include pattern recognition, image processing, artificial intelligence, neural nets, genetic algorithms, and fuzzy sets and systems.

In 1986, Dr. Pal was awarded the Fulbright Post-doctoral Visiting Fellowship to work at the University of California, Berkeley and the University of Maryland, College Park. In 1989 he received an NRC-NASA Senior Research Award to work at the NASA Johnson Space Center, Houston, Texas, U.S.A. He received the 1990 Shanti Swarup Bhatnagar Prize in Engineering Sciences. In 1993 he was awarded the Jawaharlal Nehru Fellowship, the Vikram Sarabhai Award, and the NASA Tech. Brief Award. He received the 1994 IEEE TRANSACTIONS ON NEURAL NETWORKS Outstanding Paper Award. He is a co-author of the book *Fuzzy Mathematical Approach to Pattern Recognition*, (New York: Wiley, 1986), and a co-editor of the book *Fuzzy Models for Pattern Recognition*, (New York: IEEE Press, 1992). He has written more than 190 research papers including 13 in edited books. He is listed in *Reference Asia, Asia's Who's Who of Men and Women of Achievement*, and *Biography International*. He is a Fellow of the Indian National Science Academy, National Academy of Sciences, India and the IETE; a Member of the Executive Advisory Editorial Board of IEEE TRANSACTIONS ON FUZZY SYSTEMS and the *International Journal of Approximate Reasoning*; an Associate Editor of IEEE TRANSACTIONS ON NEURAL NETWORKS, *Neuracomputing, Applied Intelligence, Information Sciences(c), Pattern Recognition Letters*,and the *Far-East Journal of Mathematical Sciences*. He is a Member of the Editorial Reviewing Board for IEEE COMPUTER MAGAZINE, the *International Journal of Applied Intelligence* and *Mathematical Reviews*; and an Executive Committee Member of the ISFUMIP and IUPRAI. He is also a Permanent Member of the INDO-US Forum for Cooperative Research and Technology Transfer (IFCRTT).