International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:2, No:6, 2008

Robust Artificial Neural Network Architectures

A. Schuster

Abstract**—** Many artificial intelligence (AI) techniques are inspired by problem-solving strategies found in nature. Robustness is a key feature in many natural systems. This paper studies robustness in artificial neural networks (ANNs) and proposes several novel, nature inspired ANN architectures. The paper includes encouraging results from experimental studies on these networks showing increased robustness.

Keywords**—** robustness, robust artificial neural networks architectures.

I. Introduction

Robustness, as forthcoming sections are going to show shortly, is relatively easy to understand via examples. Unfortunately, it is quite difficult to define. Its elusive character, so far, prevented a generally acknowledged definition. For this reason this paper adopts a view that is widely shared by saying that: “A robust system is a system that tolerates faults”. Robustness is not a new concept. It has been recognized as an omnipresent feature in many systems, natural and artificial alike, for some time. It is only recently that scientist are intrigued to study this interesting concept more rigorously and more formally. This paper aims to contribute to this effort in several ways. Its main goals are to identify fundamental strategies (e.g., redundancy, granularity, adaptation, repair, and self-healing) nature applies to make systems robust and to study their value as general design principles for AI. In order to do so, the paper has the following structure. Section II continuous with a general discussion on robustness, explaining, for example, why robustness is important in nature and how it has found its way into modern technology and AI. Section III suggests how traditional ANN architectures may be made more robust, i.e., more fault tolerant. These proposals are put to the test in Section IV and Section V. Section VI provides a discussion and Section VII ends the paper with a summary.

II. Robustness

This section intends to highlight, via a small number of examples, how robustness, as it is found in nature, appears in modern technology and AI. The observation that robustness appears on different scales and levels of complexity may be used as a starting point. For example, a single biological organism or a collective of biological organisms may show various types of robustness. A single biological organism may be called robust if it can recover from injury. For example, in case of a minor injury such as a cut on the skin, for instance, an animal may recover by growing a new layer of skin over a wound. On the other hand, a group of biological organisms may be called robust for different reasons. The collective may be called robust if the larger goals pursued by the collective remains intact or is able to recover from error. Typical examples include ant colonies or bee swarms, for instance. These collectives may still be functioning as a whole even though a number of ants or bees may loose their lives, for example, due to natural disaster or intrusion of natural foes. An interesting aspect in this regard may be the loss of the queen in a colony. In this case, the damage to the colony can be substantial. Very recently, this proneness to loss of key components in complex networks has raised considerable interest in various areas of modern technology. Network theory, for example, involves the study of so-called random networks and scale-free networks [1]. For random networks, theory expects a normal distribution between nodes and the number of connections between nodes. Basically, most nodes have an average number of connections and significant deviations from this average number are rare. On the other hand, for scale-free networks major findings show that the majority of nodes has a small number of links only and a small minority of nodes has a huge number of links. These differences in network topology influence the dynamic behavior of a network. For example, compared to random networks scale-free networks seem to be remarkable resistant or robust to accidental failures. On the flip-side, they seem to be extremely vulnerable to coordinated attacks. This behavior bears similarities to the dynamic of collectives of biological organisms described earlier. Another example, linking robustness in nature and technology quite well, is the field of information and coding theory. In nature, the genetic code was assumed to maximize efficiency and information density for some time. Nowadays the code is investigated from the point of view of providing maximum fault-tolerance or robustness [5]. Analogous, in order to make telecommunication systems fault tolerant hence robust, telecommunication technology applies error detection and error correction codes to confirm or protect information transmitted over communication channels [8]. Robustness has found its way into AI in various ways too. Genetic algorithms, for example, generate solutions to a problem from a pool of potential and similar, but usually not identical candidate solutions. If one of these candidate solutions is extremely poor a genetic algorithm may still produce an acceptable solution in the end. Genetic algorithm are also quite robust to variations in the number of potential solutions in a population. A few candidates more or less in a population usually does not affect the problem-solving potential of an application. There are many more examples, particularly amongst soft computing techniques. For instance, Schuster [11] identified robust behavior in various components of a robot control system. The components concerned include a fuzzy logic control unit, a genetic algorithm component, and a component that uses concepts derived from chaos theory. Without going into these techniques here, it is sensible to mention that robustness can be particularly important in safety critical situations or in environments that are characterized by limited human oversight. The National Space Administration

(NASA), for example, is aware of these considerations and puts robustness high on its agenda when designing system architectures for unmanned autonomous space flight systems that must complete missions with limited human control [4]. It is also important to mention that more recently robustness has risen to considerable status in the relatively young field of so-called "New AI" [3]. Of course, New AI is closely related to traditional AI and it goes without saying that the two can not be discussed in total separation from each other. The two may be separated by saying that traditional AI investigates intelligence and cognition from an algorithmic, computational point of view, whereas New AI investigates intelligence from the viewpoint of a creative interplay between one or more entities, so-called agents, and a complex, real world environment [10]. For example, chess and checkers are typical classical AI domains, whereas a project in New AI may involve a humanoid robot roaming around in a conference center assisting conference delegates. The key assumption is that robustness is directly related to the complexity of the environment in which an agent has to function. The more complex the environment, the higher the demands in terms of robustness for an agent one might say. Finally, in an AI context one thinks of software systems of course, and so, it is possible to be inclined to think about robust software. This is fine, but it is necessary to be careful. The literature and a wealth of real-life experience tell us that robustness is a problem in almost all large-scale software development projects [6]. The majority of these projects however do not contain any AI at all. Crudely, this instance of robustness may be seen as a "software engineering" problem. AI systems provide an additional layer of complexity on top of this. Consequently, making AI systems fault tolerant or robust increases the challenge significantly.

III. Robust Artificial Neural Networks

Figure 1 provides a starting point. Figure 1 illustrates a common, traditional ANN architecture with one input layer, one hidden layer, and one output layer. The network has five input neurons (attributes a_1 to a_4, plus a bias), six neurons in the hidden layer, and three output neurons (o_1 to o_3). This paper introduces various basic network architectures. Throughout, the paper refers to a traditional ANN architecture, such as the architecture in Figure 1, for example, as architecture "A_0".

From a higher level view the network in Figure 1 represents a domain. Attribute values on the input layer represent a question in this domain, and the output represents an answer generated by the network for a particular input. The question this work is interested in here is: "How robust is this network?" To answer this question imagine the following. ANNs are often regarded as simplistic models of biological brains. Assume now, that the network in Figure 1 is part of a biological brain. Based on this assumption one may be curious about what happens in a situation where one or more signals arriving at the input layer are lost (e.g., due to injury) and also, whether there are mechanisms to compensate any detrimental effects. One may assume, and rightly so, that in case of errors the decision making capacity of the network declines. How much, may depend on the severity of an error (defect, injury, loss).

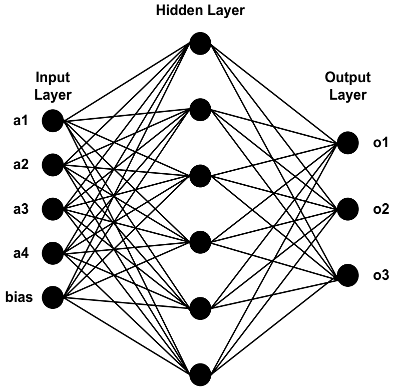

Fig. 1. A traditional ANN with network architecture type "A_0". The question is: "How robust is this network?"

In extreme cases, the network may deteriorate to a state where it is entirely useless. This study focuses on these aspects. The forthcoming sections present various solutions that have the potential to handle – to some degree – the type of error just mentioned. These solutions are not meant to outperform traditional networks (e.g., Figure 1) in terms of classification quality. The aim is to produce more robust networks, networks that can deal with the type or error mentioned before to some degree. This also means that this study is not so interested in common, potential sources for errors in ANNs, such as size of training sets and testing sets, learning rates, etc. These topics have been dealt with in the literature extensively (see [9], for example), and this study feels that its aims are fundamentally different from these. However, another aim of this study is to produce models bearing some resemblance to natural, biological neuronal networks.

A. Robust Network Architecture 1

The goal is to propose and test novel, robust ANN architectures. Essentially, the proposal involves two architectures. Both architectures are inspired by the observation that many natural, biological systems employ a safety in numbers (redundancy) strategy for handling errors. For example, an earlier section mentioned ant colonies and bee swarms where the global goal of a colony or a swarm remains intact even in cases where several members of a collective may die due to intrusion of foes or natural disaster, for example. The strategy also features in several human senses. For example, the human skin contains a considerable number of temperature sensors distributed over it. In case some of these sensors are destroyed, maybe through a minor injury, it is still possible for a person to sense temperature to a reasonably good degree. Figure 2, which represents one of the approaches investigated in this work, illustrates the application of the safety in numbers (redundancy) aspect just mentioned quite well.

For example, related to Figure 2 it is possible to think of a human skin. The input layer in Figure 2 may correspond to the skin surface with a large number of temperature sensitive nerve cells, and where nearby neurons may receive similar (in Figure 2 identical) signals. The hidden and output layer may

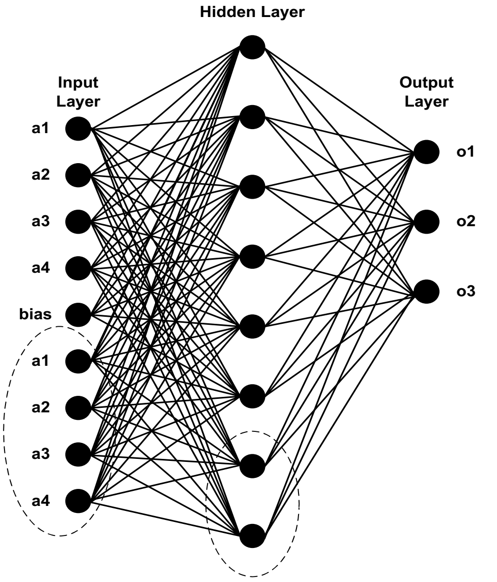

Fig. 2. Network architecture type "A_1". The network has a robustness factor of $R_{A1} = 1$

be part of a larger nervous system where signals are processed and interpreted.

In any case, a comparison between the network in Figure 2 and the network in Figure 1 reveals the following. Both networks have the usual three layers. The network in Figure 2 has two additional neurons in the hidden layer and also one additional copy of all input attributes (a_1 to a_4) in its input layer. The bias is not duplicated. The reason for this is that a bias is usually set to a constant input (e.g., the value 1.0), and so, it is possible to look at a bias as a pre-set network architecture component rather than as an external, variable input signal. Of course, the bias may have been duplicated too, but this choice, as several others, is a result of the experimental character of this study. Actually, this experimental, trial and error character is typical for many ANN studies.

Anyway, with regard to this study, a network architecture similar to that illustrated in Figure 2 is referred to as architecture "A_1". When such an architecture is used, the number of copies of a complete set of input values (excluding bias) is referred to as the robustness factor "R_{A1}" of the system. Based on this convention, the network in Figure 2 has R_{A1} = 1, and the traditional network in Figure 1 has R_{A1} = 0.

B. Robust Network Architecture 2

Figure 3 illustrates the second approach investigated in this work. Similar to the first approach, this approach features the safety in numbers strategy for error handling or robustness too.

The network in Figure 3 is quite similar to the traditional network in Figure 1. For example, input layer, hidden layer, and output layer are identical in both networks. The main difference in Figure 3 is the additional layer L^+ in front of the original input layer. This additional layer includes several copies of the original input attributes. For example, in Figure 3 an input neuron (e.g., $n1$) in the original input layer receives the original input value $a1$ plus k = 4 additional copies of $a1$. Note that the bias is not duplicated again. In terms of processing the following happens. For every attribute (e.g., $a1$) the value arriving at a neuron in the original input layer (e.g., $n1$) is calculated by taking the average of the sum of the $k+1$ attribute values arriving on layer L^+. For example, imagine an attribute value of $a1$ = 0.240 in Figure 3, and where $k = 4$. In this case, the value arriving on $n1$ calculates to: $\frac{(k+1)*0.240}{(k+1)} = \frac{5*0.240}{5}$ = 0.240. The example indicates that in the absence of error the network in Figure 3 behaves like the network in Figure 1. In case of error however, there are differences. For example, imagine two of the five $a1$ inputs in layer L^+ in Figure 3 receiving an erroneous value of "0.0". In this case, taking the value $a1$ = 0.240 again, the value arriving on $n1$ amounts to $\frac{(k+1-2)*0.240}{(k+1)} = \frac{3*0.240}{5}$ = 0.144.

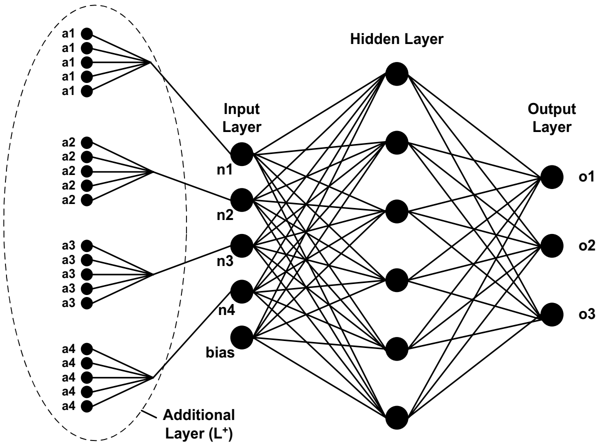

Fig. 3. Network architecture type "A_2". The network has a robustness factor of $R_{A2} = 4$

Here, a network architecture similar to that illustrated in Figure 3 is referred to as architecture "A_2", and the number of "additional" copies for a particular input variable (excluding bias) as the robustness factor "R_{A2}" of the system, respectively. Based on this convention, the network in Figure 3 has R_{A2} = 4, and the original network in Figure 1 has R_{A2} = 0.

IV. Data And Testing Procedures

The aim is to compare network architectures A_0 (Figure 1), A_1 (Figure 2) and A_2 (Figure 3) in terms of robustness. This section outlines the experimental setup and other details behind the various studies undertaken in this work.

Test Data: All studies use Fisher's Iris Plant data set[1]. This data set is well-known in the research community. It contains 150 records. Each record is described by four attributes (sepal length, sepal width, petal length, and petal width) of one of three iris classes (Iris-setosa, Iris-versicolor, and Iris-virginica). Every class contains 50 records. One class is linearly separable from the other two classes, but the other two classes are not linearly separable from each other.

[1] Source: Blake C.L., and Merz C.J.: UCI Repository Of Machine Learning Databases, University of California, Irvine, Dept. of Information and Computer Sciences, 1998, http://www.ics.uci.edu/~mlearn/MLRepository.html.

General Testing Procedures: There are three main studies: Study-0, Study-1, and Study-2. Study-0 investigates architecture A_0, Study-1 applies architecture A_1, and Study-2 concentrates on architecture A_2. Each study contains several tests. Every test uses an ANN with one of the three main architectures underlaying. In a test a network is trained and tested according to the well-established “leave-one-out cross-validation” procedure. In leave-one-out cross-validation a network is trained with all records in a data set excluding “one” record (here, $150 - 1 = 149$ records). In network training, a single, sequential presentation of all 149 records to a network is called an “epoch”. All networks are trained over 200 epochs. After training, the record left out in the training process is tested for classification. Classification uses standard ANN procedures such as binary representation for classes, and thresholds for class membership, for example. This process is conducted for every record in the data set, and the complete procedure just explained is called a “cycle”. In order to avoid one-off outcomes, results are averaged in every study over 100 cycles. Consequently, results generated in a single test correspond to: 100 cycles $*$ 150 records = 15000 individual classification tests.

Strategy: Initially error-free networks are tested, and then networks with increasing, artificially induced error rates. For example, all tests train a network with 150 - 1 = 149 original, always error-free Iris Plant records. A trained network is tested on one record, the record left out in the training process. In an error-free test this record is unchanged. For example, one of the Iris Plant records has the following set of values $\{0.222, 0.624, 0.067, 0.041\}$. These values are normalized in the interval [0, 1], which is typical for ANN applications. In an error-free test a network tests this record for classification. In contrast, tests involving erroneous data model an error in two ways. One approach models a “systematic” error by replacing one or more values in a record with the value “0.0”. A second approach creates a “random” error by always replacing one or more values in a record with a value drawn from the interval [0, 1]. A random selection procedure decides in both approaches which values in a record are affected by an error. For example, a single, systematic error may replace the second value in the aforementioned record with the value 0.0, producing the erroneous record $\{0.222, \mathbf{0.0}, 0.067, 0.041\}$. Here, the error affects one of four values, which is equivalent to an error rate of 25%. Finally, because of the special demands of the various ANN architectures, all programs in this study are hand-coded using the commercial Delphi programming software.

V. Results

The forthcoming sections present results for Study-0, Study-1, and Study-2.

A. Results Study-0

Study-0 examines the robustness of the traditional ANN in Figure 1. Study-0 creates errors of 0%, 25%, 50%, 75% and 100%, affecting 0, 1, 2, 3, or all 4 values in a record, respectively (the bias is never affected). Table I and Figure 4 illustrate how the network copes with these errors.

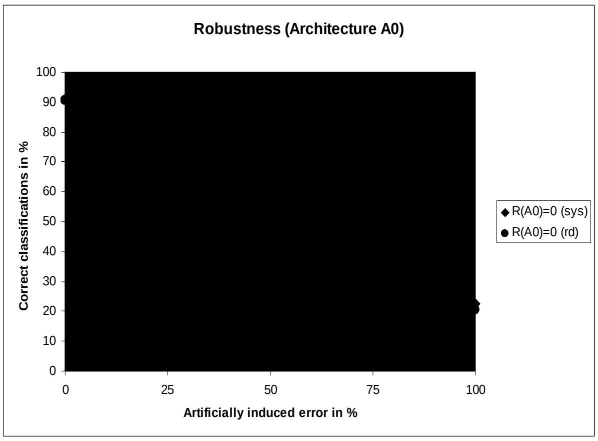

Fig. 4. Performance graph for the traditional network in Figure 1. A single dot represents 15000 individual tests

Table I and Figure 4 illustrate that the error-free network classifies around 91% of all test records correctly (90.9% systematic error and 90.7% random error). The reliability of these results was verified by applying the commercial data mining tool DataEngine V.2.0 to the same problem. It was found that the results generated by the tool are in line with the results in this paper. Overall, Figure 4 illustrates that the network performance decreases with increasing error rates. For example, a systematic error of 25%, which corresponds to always setting one of the four values in a test record to the value 0.0, brings classification correctness down from 90.9% to 47.7%, which is a significant drop. Note that the expression (1) next to the value 47.7% in Table I indicates that one of four values (i.e., 25%) in a test record was affected by an error. In contrast, a random error of the same magnitude reduces classification correctness down to 59.5%, which is a quite significant drop of more than 30%. For maxim error (100%) both types of error converge towards values in the lower 20% range.

The main observation for Study-0 is that a traditional network architecture can be quite vulnerable to error. Even an error affecting a single input neuron only, may impact network performance severely. Study-0 also indicates that there may be differences between systematic errors and random errors. The network seems to react better to random errors rather than to systematic errors. The forthcoming sections query whether network architectures A_1 and A_2 can do better than this.

B. Results Study-1

Before the start of this analysis consider the expected outcome in this study. Study-0 found that network performance breaks down quite significantly around the 25% error region (see Figure 4). For errors larger than this it is possible to label the network unreliable. Consequently, two things may be expected from Study-1. First, compared to the traditional network in Study-0, the network in Study-1 should react more positively (robust) to errors in general. Second, a positive

TABLE I
STUDY-0 RESULTS. A_0 = UNDERLYING NETWORK ARCHITECTURE (FIGURE 1); I, H, O = NEURONS IN INPUT, HIDDEN, OUTPUT LAYER; R_{A0} = ROBUSTNESS FACTOR, SYSTEMATIC AND RANDOM ERROR

	A_0	Error [%]				
R_{A0}	**I, H, O**	**0.0**	**25.0**	**50.0**	**75.0**	**100.0**
0, Systematic	5, 6, 3	90.9 (0)	47.7 (1)	34.6 (2)	26.7 (3)	22.4 (4)
0, Random	5, 6, 3	90.7 (0)	59.5 (1)	39.7 (2)	27.9 (3)	20.4 (4)

reaction for errors up to the 25% mark is particularly welcome. Larger errors are considered as quite severe again, in which case the network can not be trusted again.

The paper now refers to the results for network architecture A_1 in Study-1, which are given in Table II, Figure 5, and Figure 6. Note that the data provided here is a subset of the data generated overall. This data therefor stands representatively for the presented work. Table II holds results for architecture A_1 networks, robustness factors R_{A1} of 1 and 5, and systematic errors and random errors. Figure 5 and Figure 6 split the data in Table II. Figure 5 illustrates the results for systematic errors and Figure 6 those for random errors only. For clarity, both figures connect data points via help-lines. To ease comparison, the figures also include the so-called "base lines" $R_{A0} = 0$ (sys) and $R_{A0} = 0$ (rd) from Figure 4 (dotted lines).

Figure 5 illustrates that even the smallest robustness factor, $R_{A1} = 1$ (sys), immediately contributes to classification correctness. Up to the 50% mark, $R_{A1} = 1$ (sys) is always clearly above base line $R_{A0} = 0$ (sys). In the same region, $R_{A1} = 5$ (sys) dominates the base line even more clearly.

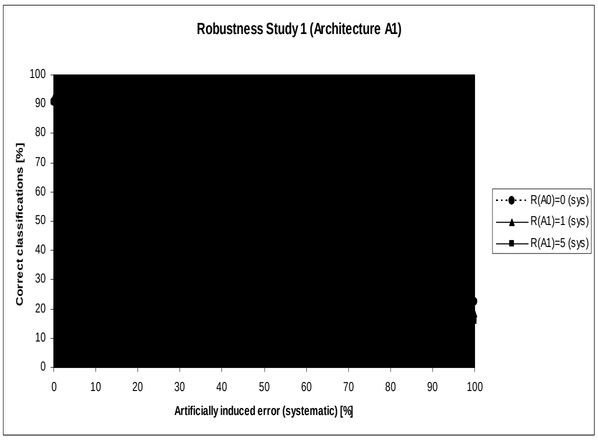

Fig. 5. Study-1 performance graph; systematic error

An general observation is a significant performance improvement associated with robustness factor $R_{A1} = 5$ (sys) in the 25% range. For example, for an error of 25%, $R_{A1} = 5$ (sys) improves performance by 20% from 47.7% to 67.7%. For a 12.5% error the network does even better, achieving 83.3%, which is relatively close to the error-free 90.9% in Study-0. Here it is important to mention that in Study-1, for $R_{A1} = 5$ (sys), a 12.5% error affects 3 of 24 inputs (see Table II). This is important, because in Study-0 an error affecting three inputs is equivalent to an error-rate of 75%. This study therefore identifies a second, very important concept related to robustness, namely the concept of "granularity". Figure 6 illustrates similar results for random errors.

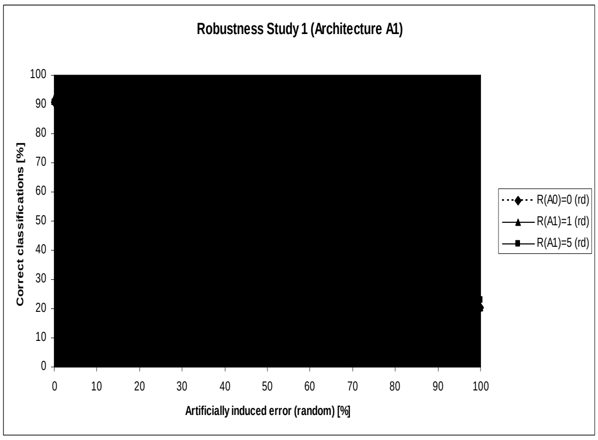

Fig. 6. Study-1 performance graph; random error

For example, Figure 6 illustrates immediate improvement for $R_{A1} = 1$ (rd). In the critical 25% error range $R_{A1} = 5$ (rd) indicates the desired, significant performance improvement. For example, for an error-rate of 25%, $R_{A1} = 5$ (rd) improves performance by 17.7% from 59.5% to 77.2%. For 12.5% error the network achieves 86.9%, which is quite close to the error-free 90.7% in Study-0.

The main findings in Study-1 are as follows. Study-1 indicates differences between systematic errors and random errors again. The network seems to react better to random errors rather than to systematic errors. The most important finding however is that the approach behind Study-1 leads to significant performance improvements and hence significantly more robust networks in general. This is an important finding!

C. Results Study-2

Table III, Figure 7, and Figure 8 illustrate results for Study-2. Table III holds results for architecture A_2 networks, robustness factors R_{A2} of 1, 10, and 100, and systematic and random errors. Figure 7 and Figure 8 illustrate a subset only of the Table III data. Both figures contain their corresponding base line (dotted lines) from Figure 4. Data points are connected by help-lines again.

In many ways Study-2 repeats the findings in Study-1. For example, Figure 7 illustrates the immediate, positive impact on classification correctness provided by a robustness factor of one, i.e., $R_{A2} = 1$ (sys).

Larger robustness factors lift performance up to even higher values. For example, for a systematic error of 25%, $R_{A2} = 100$

TABLE II

STUDY-1 RESULTS. A_1 = UNDERLYING NETWORK ARCHITECTURE (FIGURE 2); I, H, O = NEURONS IN INPUT, HIDDEN, OUTPUT LAYER; R_{A1} = ROBUSTNESS FACTOR, SYSTEMATIC AND RANDOM ERROR

		A_1		Error in % and neurons affected								
R_{A1}	**I**	**H**	**O**	**0.0**	**12.5**	**25.0**	**37.5**	**50.0**	**62.5**	**75.0**	**87.5**	**100.0**
1, Systematic	9	11	3	92.4 (0)	71.8 (1)	57.8 (2)	49.6 (3)	40.4 (4)	32.9 (5)	25.7 (6)	21.1 (7)	18.4 (8)
5, Systematic	25	27	3	90.7(0)	83.3 (3)	67.7 (6)	51.6 (9)	38.2 (12)	26.2 (15)	22.9 (18)	19.6 (21)	15.9 (24)
1, Random	9	11	3	92.4 (0)	79.6 (1)	66.4 (2)	55.0 (3)	44.2 (4)	36.0 (5)	29.5 (6)	25.4 (7)	20.5 (8)
5, Random	25	27	3	90.7 (0)	86.9 (3)	77.2 (6)	64.5 (9)	49.2 (12)	38.2 (15)	30.3 (18)	25.9 (21)	22.9 (24)

TABLE III

STUDY-2 RESULTS. A_2 = UNDERLYING NETWORK ARCHITECTURE (FIGURE 3); I, H, O = NEURONS IN INPUT, HIDDEN, OUTPUT LAYER; R_{A2} = ROBUSTNESS FACTOR, SYSTEMATIC AND RANDOM ERROR

		A_2		Error in % and neurons affected								
R_{A2}	**I**	**H**	**O**	**0.0**	**12.5**	**25.0**	**37.5**	**50.0**	**62.5**	**75.0**	**87.5**	**100.0**
1, Systematic	9	6	3	90.9 (0)	70.0 (1)	56.6 (2)	49.4 (3)	41.6 (4)	34.4 (5)	27.1 (6)	23.7 (7)	22.5 (8)
10, Systematic	45	6	3	90.9 (0)	80.4 (6)	67.4 (11)	53.9 (16)	40.4 (22)	31.7 (28)	27.5 (33)	25.5 (38)	22.4 (44)
100, Systematic	405	6	3	90.9 (0)	83.9 (50)	67.2 (101)	53.3 (152)	38.3 (202)	31.7 (252)	29.2 (303)	25.8 (354)	22.5 (404)
1, Random	9	6	3	90.4 (0)	77.3 (1)	64.4 (2)	52.3 (3)	43.1 (4)	34.5 (5)	28.5 (6)	23.9 (7)	20.7 (8)
10, Random	45	6	3	90.5 (0)	85.7 (6)	75.3 (11)	60.2 (16)	44.4 (22)	35.2 (28)	31.7 (33)	30.3 (38)	29.1 (44)
100, Random	405	6	3	90.6 (0)	88.6 (50)	80.2 (101)	60.0 (152)	41.2 (202)	33.7 (252)	33.1 (303)	33.1 (354)	33.0 (404)

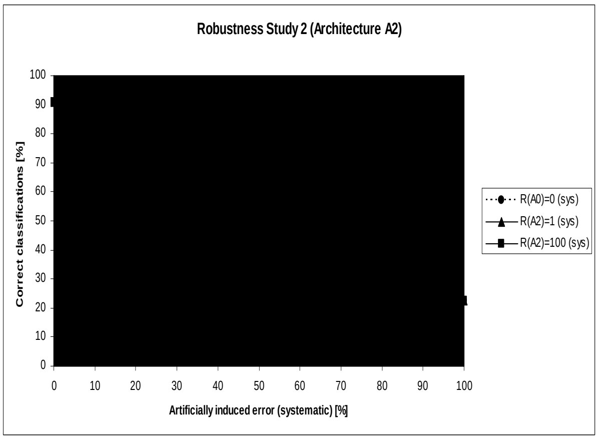

Fig. 7. Study-2 performance graph; systematic error

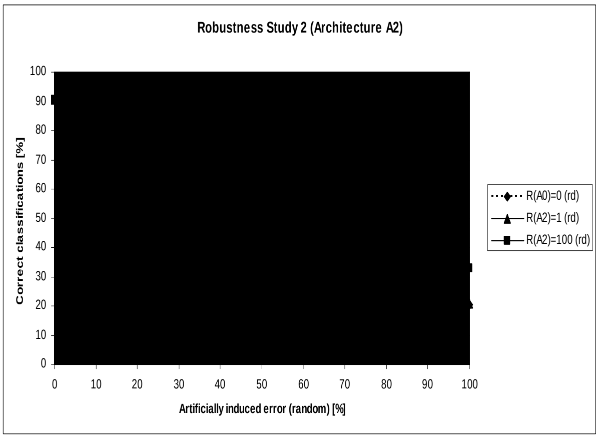

Fig. 8. Study-2 performance graph; random error

(sys) lifts classification correctness by 19.5% from 47.7% to 67.2%. For a systematic error of 12.5%, classification correctness goes up to 83.9%, which is not too far away from the 90.9% in the error-free network in Study-0. As in the previous studies, Study-2 also reveals a difference in behavior between systematic errors and random errors. Table III and Figure 8 illustrate that A_2 networks seem to cope better with random errors again.

For example, for a random error of 25%, $R_{A2} = 100$ (rd) improves classification correctness by 20.7% from 59.5% to 80.2%. For a random error of 12.5%, classification correctness goes up to 88.6%, which is really close now to the 90.7% in the error-free network in Study-0.

A summary for Study-2 is similar to the findings in Study-1. There are differences between systematic errors and random errors, and the network seems to react better to random errors again. The most important finding in Study-2 is that the approach behind the study leads to significant performance improvements and hence significantly more robust networks.

VI. DISCUSSION

The results in this paper are quite positive overall. They indicate that, for particular errors, it is possible to significantly increase the robustness of traditional ANN architectures. The work presented in the paper shows that this gain in performance is due to specific network topologies. These topologies are inspired by design principles found in nature, a finding which is crucial for this work. The paper identifies two major design principles at work; "redundancy" is the obvious one, but there is also "granularity". The two principles seem to stand in an inverse relationship. For example, a characteristic of Study-1 are relatively small robustness factors (low redundancy), but the study involves larger networks (high granularity) to train. Study-2, on the other hand, always uses the basic network structure in Figure 1 (low granularity), and seems to attain its robustness via the many (high redundancy) additional copies of its input values. The redundancy versus granularity interplay is not a new discovery. It can be found in many shapes in biology, but also in technology. An example in biology may be a pride of lions and an ant colony. Lions are larger (granularity) beasts than ants, but an ant colony contains many more individuals (redundancy). The interplay between granularity and redundancy however makes both groups robust in the sense of survival, and so, the granularity-redundancy interplay may be viewed as a survival strategy. An analogous example in technology is software design, which is an area where other, related terms such as “high cohesion” and “low coupling” also play important roles [7]. It may be important again to emphasize the difference in which robust topologies accommodate minor errors. In the traditional network the smallest possible error, the failing of one single input, already represents an absolute error of 25.0%. Alternatively, for Study-2 A_2 topologies with robustness factors of 100, failure of one single neuron represents an absolute error of about one percent only. From this perspective, it is worthwhile mentioning that the presented work addresses the IT philosophy of “graceful degradation” [2]. For instance, compared to traditional ANN architectures, robust ANNs behave more stable in case of minor errors and degrade more gracefully when larger errors occur. Finally, despite an awareness that the presented study is specific, it is fair to suggest that the work has some general value for AI. For example, autonomic computing, ambient computing, and pervasive computing all entertain the analogy between (hugely complex) intelligent systems equipped with a multitude of sensors, actuators, etc. and biological brains. With some imagination it may be possible to view the ANNs in the various studies as mini brains (systems) where each attribute going into an ANN may be a sensor reading. The presented work contributes to the robustness of such systems, but how the approach extrapolates to larger systems is an open question at this stage.

VII. Summary

This paper investigated robustness as it is found in nature as a design principle for AI. The paper provided a discussion on robustness as it is found in nature and highlighted how robustness appears in modern technology and AI. The paper then studied the robustness of several novel ANN architectures. When compared to traditional ANN topologies the paper found that these novel topologies can be significantly more robust. A discussion reflected on the presented work in a wider context. Current work studies robustness more comprehensively in other AI areas.

References

[1] A.L. Barabási and E. Bonabeau, “Scale-free networks,” *Scientific American*, vol. 288, pp. 55–59, May 2003.

[2] P. Bentley, “Investigations into graceful degradation of evolutionary developmental software,” *Natural Computing*, vol. 4, pp. 417-437, 2005.

[3] H. Brighton and H. Selina, *Introducing Artificial Intelligence*. Icon Books UK, 2003.

[4] D. Dvorak, G. Bollella, T. Canham, V. Carson, V. Champlin, B. Giovannoni, M. Indictor, K. Meyer, A. Murray, and K. Reinholtz, “Project golden gate: Towards real-time Java in space missions,” in *Proc. 7^{th} IEEE International Symposium on Object-Oriented Real-Time Distributed Computing (ISORC 2004)*, IEEE Computer Society, Vienna, Austria, 2004, pp. 15–22.

[5] B. Hayes, “Ode to the code,” *American Scientist*, vol. 92, pp. 494–498, November-December 2004.

[6] M.N. Huhns and V.T. Holderfield, “Robust software,” *IEEE Internet Computing*, vol. 6:2, pp. 80–82, March-April 2002.

[7] D. Jackson, “Dependable software by design,” *Scientific American*, vol. 294, pp. 58–65, June 2006.

[8] G.A. Jones and M.J. Jones, *Information and Coding Theory*. Springer Verlag London, 2000.

[9] K. Mehrotra, C.K. Monan, S. and Ranka, *Elements of Artificial Neural Networks*. The MIT Press, 1997.

[10] R. Pfeifer, “Embodied artificial intelligence: trends and challenges,” in *Embodied Artificial Intelligence*, Lecture Notes in Computer Science LNCS, vol. 3139, F. Iida, R. Pfeifer, L. Steels, and Y. Kuniyoshi, Eds. Springer-Verlag Berlin, Heidelberg, New York, 2003, pp. 1–26.

[11] A. Schuster, “A hybrid robot control system based on soft computing techniques,” in *Advances in Applied Artificial Intelligence*, Lecture Notes in Artificial Intelligence LNAI, vol. 4031, A. Moonis and D. Richard Eds. Springer Verlag, Berlin, Heidelberg, New York, 2006, pp. 187–196.

Alfons Schuster is a Lecturer in the School of Computing and Mathematics at the University of Ulster at Jordanstown, in Northern Ireland. He was awarded a DPhil in Computer Science from the University of Ulster in 1998, and a BSc in Applied Physics from Munich University of Applied Sciences in Munich, Germany, in 1990. He has worked in industry for several years as an engineer. His research concerns artificial intelligence, robotics, soft computing, theory of computing, and natural computing. He has published 43 research papers in these areas.