Microcomputers in Civil Engineering 9 (1994) 97–108

Effect of Representation on the Performance of Neural Networks in Structural Engineering Applications

D. J. Gunaratnam & J. S. Gero

Department of Architectural and Design Science, University of Sydney, NSW 2006, Australia

Abstract: *The pattern-mapping, pattern-classification, and optimization capabilities of neural networks have been used to solve a number of structural analysis and design problems. Most applications exploit the pattern-mapping capability and are based on the back-propagation paradigm for neural networks. There are a number of factors that influence the performance of these networks. This paper initially discusses these factors and the domain-dependent and -independent techniques presently available for improving performance. The paper then considers the effect of representation, selected for the input/output pattern pairs, on the performance of these networks and demonstrates that representations based on dimensionless terms, derived from dimensional analysis, lead to improved performance. It is shown that dimensional analysis provides a representational framework, with reduced dimensionality and embedded domain knowledge, within which effective learning can take place and that this representational change can be used to enhance the domain-independent and -dependent techniques presently available for improving performance of these networks.*

1 INTRODUCTION

A number of machine learning paradigms have been used in the past to solve structural analysis and design problems. These paradigms were implemented using either a symbolic, connectionist, or genetic algorithm approach. Of these, the connectionist approach has, of late, attracted considerable interest, and the results of a number of applications in the structural analysis and design area have been published recently.[1,2,11–15,18,19,21,29] There are also publications in the related application area of material modeling,[4,8] which can represent the initial stage in the structural analysis solution process.

In neural networks or connectionist models of computation, attempts are made to simulate the powerful pattern-recognition capabilities of the human brain and to use this capability to represent and manipulate knowledge in the form of patterns. The recent resurgence in neural network research and the consequential increase in its applications have been due to the development of new neural network architectures and powerful learning algorithms. A variety of neural network models now exist, each with its own special characteristics and capabilities, which make them suitable for different types of applications.

In the area of structural engineering, the multilayer feed-forward networks, based on the supervised learning procedure and the back-propagation learning algorithm, have been used extensively. These networks have been trained to provide response information given a structural description; the required geometric parameters, when only partial structural descriptions are available; and the element design parameters, for both preliminary and optimal designs, given the geometric and topologic design information for the structure. They also can provide material models, synthesized either from experimental observations or, as in the case of composites, from basic information relating to the environment and the constituents of the composites.

The rationale for selecting a connectionist approach in most of these applications is that once the network has been trained, it can then provide the required analysis or design information at a fraction of the computational effort required to generate the same information using the well-established approaches based on causal models. For most of the applications reported in the literature, these algorithmic approaches were used mainly to generate the training sets for the networks, and in all these cases, the time taken to

train the network was much greater than the time taken to generate the training set. This approach is thus viewed as a means of encoding and extending the knowledge gained from previous analysis and design experiences.

Two other types of network models also have been used to solve structural engineering problems. One of these, the Hopfield network, has the capability for recall, even when the input patterns are fuzzy or not fully defined, and can hence function as an associative memory. The stable states of this network, which are used to store information for retrieval, are associated with the minima of an energy function, and this attribute has been exploited to solve combinatorial structural optimization problems.[14,15] The other network model, known as the ART (adaptive resonance theory) network, is based on the unsupervised learning procedure and can be used to classify input patterns into specific categories, and this capability is adapted to solve structural analysis and design problems.[13–15]

There are a number of factors that influence the performance of these networks. These factors are determined not only by the features of the network but also by the representations used to describe the patterns that the network learns to map. The representations selected for the input/output patterns determine some of the topologic features of the network as well as the level of complexity of the mapping to be learned and hence the complexity of the network's internal representation of the mapping. Representation plays an important role in human problem solving and in learning.[28] It is therefore to be expected that representation would play an equally important role in machine learning applications as well. The input/output pattern pairs usually can be described in terms of a set of associated vector pairs. By pre- and postcoding of the input and output vectors, respectively, it is possible to modify the way any given network would generalize and hence its performance.[17]

Performance has been a major concern in most of the structural engineering applications, particularly the generalization capabilities of the networks. In solving structural optimization problems, the trained network is repeatedly used within the optimization process to provide analysis and sensitivity information. If the trained network performs poorly in the generalization tasks, the design solutions obtained by this process can be far from the optimal.[2,12] This possibility for gross errors in final results also can occur in the use of material models that are obtained from trained neural networks and used repeatedly within a finite-element analysis to generate element characteristics.

The present investigation is hence directed toward using the learning capabilities of the network to generate high-level relationships—not previously known—for structural design rather than in training networks to replace well-understood algorithmic approaches. These functional relationships are useful for making design decisions, particularly at the preliminary design stage, and provide a means of capturing previous design and analysis experiences in a concise manner. A number of these high-level relationships, generated using human rather than machine learning capabilities, have been reported in the literature. Three such relationships and their derivations are discussed in refs. 7, 9, and 10.

Generating high-level relationships for structural design can be posed as a problem of search, where the search is conducted in two different spaces. Initially, a search for a suitable representational framework is conducted in the design parameter space, and this is then followed by a search of the functional space. Dimensional analysis provides the basis for conducting the first search and defines dimensionless parameters from which the required representational framework can be constructed. The search in the functional space can be based on statistical techniques, or the function can be synthesized from transfer functions and connection weights of neural networks. The present investigation uses the latter approach.

The work reported in this paper represents the initial stages in the investigation and is concerned with studying the effect of representation on the performance of multilayer feedforward networks in structural engineering applications. It demonstrates that representation plays an important role in the learning process and that dimensional analysis provides a suitable representational framework within which effective learning can take place.

The paper thus initially considers the various neural network models and identifies the features that influence their performance. It then looks at some of the domain-independent techniques available for improving the performance of multilayer feedforward networks. This is followed by a review of techniques used for improving performance within the structural engineering applications reported in the literature. It finally demonstrates that dimensional analysis provides the required domain knowledge and that dimensionless parameters can be used to construct the knowledge-dependent representational framework within which learning is to take place.

The effectiveness of this representation on neural network performance is demonstrated by comparing the results reported previously[29] with those obtained following a change of representation of the input–output pattern pairs. Three types of problems were considered, and for all three, closed-form solutions are available. The solution to the third problem, however, is in the form of an infinite series. In all three cases, the changes in representations are selected based on dimensional analysis and the mathematical models, which are known for all three cases considered.

2 NEURAL NETWORK MODELS

A number of neural network models with different characteristics and capabilities have been developed and applied successfully to a variety of tasks. The capabilities of the

different network models are determined by their structure, dynamics, and learning methods.[22] Performance of neural networks is influenced by these three aspects, and each of these aspects, in turn, is defined by a number of parameters that can be controlled in any application.

The structure of the neural network can be described at three levels. The lowest level of the network structure is described in terms of the characteristics of the processing element and is referred to as the *microstructure*. At this level, the processing elements are characterized by their summation and transfer functions, which operate on the incoming signals to produce a single output. The transfer functions commonly used are threshold logic, hard-limiting, sigmoid, tanh, and radial basis functions. In some processing elements, the input signals are modified by applying a threshold function to provide a bias term and a gain term. The bias term can have either a fixed or adjustable value, and the gain term can be in the form of an adaptable parameter that influences the learning rates. For applications with temporal sequences in the input patterns, either memory functions can be encoded in the processing elements or the summation function can be selected to include posttransfer function activation of the previous cycle.[22]

The next level, referred to as the *mesostructure*, provides a description of the organization and arrangement of the processing elements in the network. It is at this level that neural networks can be differentiated and formed into classes. Description at this level provides information on the number of layers, number of processing elements per layer, types of connections, and degree of connectivity. Based on these features, the available networks can be classified into the five different and structurally related classes: multilayer feedforward, single-layer laterally connected, single-layer topologically ordered, bilayer feedforward/feedback, and multilayer cooperative networks. In structural engineering applications, the multilayer feedforward (back-propagation), single-layer laterally connected (Hopfield), and bilayer feedforward/feedback (ART) networks have been used. In addition to these features, the networks also can differ in the types of input and output patterns they can represent and process.

A number of neural networks may have to be integrated to form a system of neural networks in order to solve complex problems. Description of the modifiable features of these system of networks is referred to as the *macrostructure*. The structural engineering applications so far have been based on the preceding network models and were hence concerned with the features at the micro- and mesostructure levels only.

The way in which a neural network processes data or the patterns presented to it is referred to as *network dynamics*. Processing could be carried out in a synchronous mode by all the elements in a layer, with information flowing in the forward direction, as in the case of a multilayer feedforward networks. The elements process the data only once as they flow through the network. In the Hopfield and ART networks, because of the nature of the connections between the processing elements, processing can be recurrent and asynchronous and continues until a stable state of the network is reached. Stability of the network dynamics is thus an important consideration in these types of networks. These networks can be shown to be globally stable if an energy function that satisfies certain criteria can be defined. The stable states of the networks then correspond to the minima in the energy surface defined by this function, and the network dynamics correspond to movement along the energy surface toward a minimum point. The structure of the network together with its dynamics determines the functional abilities of the network and hence its possible applications.

The two main learning methods used to train networks are the supervised and unsupervised learning procedures. In the supervised learning procedure, the desired output response is presented to the network for each input stimulus provided by the training set. The network uses the feedback error information between its own output and the desired value to iteratively adjust its connection weights and hence gradually organize itself to achieve the desired input/output mapping. Learning results as a consequence of the change or adaptation to the weights on the connections to minimize an error function. The changes to the connection weights are derived from a learning rule, which is generally a variation of either the Hebbian, delta, or competitive learning rule. The learning procedure is implemented through a suitable learning algorithm. In the unsupervised learning procedure, the network is presented with the input stimuli only. The network organizes itself internally to create categories by correlating the information available in the input data. Structural engineering applications that require networks to learn mappings between input/output pattern pairs have been based mainly on the supervised learning procedure and the generalized delta rule of learning.

The design of a network for a given application requires initially selecting the structure, dynamics, and learning method for the network. All structural engineering applications reported so far have been based on existing models, and their performance in these applications can be improved by controlling the modifiable features of the networks. In the next two sections, domain-independent and -dependent techniques for achieving improved performance are considered.

3 IMPROVING PERFORMANCE OF NEURAL NETWORKS

The performance of multilayer feedforward networks can be discussed in terms of the speed of learning and generalization capabilities of these networks. The speed of learning can be expressed either as CPU time or as the number of epochs required for convergence of the back-propagation

algorithm and thus can form the basis for comparison.[6,26] There is at present no formal definition of what it means to generalize correctly, but the generalization capability of the network may be assessed based on how well it performs on the test data set. These two aspects of performance can be improved by modifying the structure, dynamics, training set and schedule, learning rules, and representation selected for the input data.

The modifications to the structure of the network can be made at both the micro- and mesostructure levels. At the microstructure level, improvements in performance have been effected by rescaling the transfer function, using the radial basis functions for the hidden units, using higher-order connections, and using functional-link networks. The effects of these on the performance of the networks are discussed in ref. 22.

At the mesostructure level, the performance is improved by controlling the network parameters such as the number of layers, the number of processing elements per layer, and the degree of connectivity. Guidance is available for selecting each of these parameters. It has been shown that one hidden layer is sufficient to compute arbitrary decision boundaries for the outputs, and two hidden layers are sufficient to compute an arbitrary output function of the inputs.[22] For continuous inputs, however, it is possible to achieve good performance by using either a single or no hidden layer, provided suitable microstructure characteristics are selected or the mapping to be learned belongs to a particular class of functions.

In determining the mesostructure of the network, it is possible to select the number of input and output units as equal to the number of data items in the input and output vectors, respectively, if a local representation is assumed. It will be demonstrated in subsequent sections that the selection of input and output vectors and their representations can considerably influence performance. Rules and heuristics are available for the selection of the number of hidden units, and procedures also exist for automatic sizing of hidden layers, including provisions for adding or removing hidden units to improve performance.

Selecting the optimal number of hidden units can improve both the speed of learning and the generalization capability of the network. Removing redundant units will reduce the number of connections and hence the number of weights to be adapted during learning. It has been estimated empirically that for back-propagation the learning time on a serial machine is very approximately $O(N^3)$, where N is the number of weights in the network.[17] A reduced number of hidden units also can prevent the network from internally configuring itself to function as a table lookup scheme during the training phase and hence performing poorly in the generalization aspects of the task.[27]

The basic back-propagation algorithm, which uses the gradient-descent method to minimize the error function, is too slow for many applications and scales up poorly for larger and complex tasks. A number of faster-learning variations of the back-propagation algorithm have been developed.[6,26] Substantial improvements have been achieved by appropriately tuning the back-propagation learning parameters, eliminating flat spots due to the derivative of the sigmoid function approaching zero at the extreme values, which results in the output units getting stuck in the zero state, and improving the optimization technique, using some form of second-order weight update method or the second derivative of the error function.[6,16,26] Parallel back-propagation neural networks learning algorithms have been developed employing the vectorization and microtasking capabilities of vector MIMD machines, resulting in a significant increase in the speed of learning.[19]

The learning rate, the momentum factor, and the range of the random initial weights are the learning parameters that influence performance of the back-propagation algorithm. In the basic back-propagation algorithm, these parameters are initially set to suitable values and are held constant during the training process. The faster variations of this algorithm improve their performance by modifying the learning rate and the momentum factor, either according to a fixed training schedule or dynamically using second-order techniques.[6,26] Previous investigations indicate that adaptive changes to the learning rate, based on information derived from the error surface, can result in substantial reduction to the learning time.[6]

Problems associated with flat spots at the extremes of the sigmoid function in the output units have been eliminated and dramatic improvements to training times have been achieved simply by adding a constant value to the derivative of the sigmoid function.[26] This problem is avoided in the hidden units by normalizing and scaling the summed inputs to the units. Substantial improvements, however, occur only when adjustments are made to the output units.[6] The improved optimization techniques used have been based either on gradient information and the line-search methods or on the second derivative of the error function, usually obtained by some approximate method to avoid excessive computation.[16]

The improvements discussed so far are based on techniques that are domain-independent. Further improvements in performance can be effected by selecting a suitable training set and training schedule. In selecting the training set, the representation and composition of the input and output vectors, as well as the number and distribution of the training patterns, need to be considered. It is in the representation and selection of the input and output vectors that domain knowledge can be introduced to improve performance.

4 STRUCTURAL ENGINEERING APPLICATIONS

The neural network applications in structural engineering that have been reported so far fall into three groups: material

modeling, analysis, and optimal design. In the material modeling applications, the network learns relationships either between the stresses and strains or between the properties of the constituents and that of the composite.[4,8] In the analysis applications, relationships between geometric and section properties and the response quantities are learned.[2,12,29] In the optimal design applications, the relationships of interest have been those between the geometric parameters and section properties.[2,12]

In all these applications, attempts were made to improve performance of the network by using some of the methods discussed in the preceding section. At the microstructure level, the performance was improved by tuning the learning parameters, but this, however, was not done in an adaptive manner. Functional-link networks were introduced but did not produce the anticipated improvement in performance.[12] At the network or mesostructure level, all applications used either one or two hidden layers and selected the number of units in these layers either based on heuristics or by experimenting with different numbers of the units. The basic back-propagation algorithm was used in all applications except in ref. 21, where the Davidon-Fletcher-Powell method was used to improve the speed of learning. The number of units in the input and output layers was selected by partitioning the variables that characterize the system into independent and dependent sets. This partitioning is not unique and depended on the mapping to be learned by the network.

It is in selecting the training set and the training schedule that different approaches were adopted to improve the performance of the network. In ref. 12, the input/output training pairs were initially selected at random, either in the range defined by the upper and lower bounds of the design variables or in a region about the known optimum. In order to improve performance further, a data-clustering approach was then adopted. Five cluster centers, including centers about the optimum, were defined for the problem, and five networks were then trained based on these clustered data. This essentially partitions the space to be mapped and hence reduces the complexity of the mapping. Improved performance, both in terms of speed of learning and generalization capability, has been reported when the training set was selected based on orthogonal tables.[21] In the material modeling application in ref. 8, the network did not converge when trained with all the stress/strain paths at once. The performance of the network improved and it converged when the data were presented to the network in stages, even though no changes were made to the mesostructure. This approach is referred to as *shaping*, where the efficiency of learning is improved by increasing the training data set in batches, and is applicable to other applications as well.[22]

Most of the applications in structural analysis and design that were reported have trained the networks using representations that were used to generate the training and testing sets, and only a limited attempt was made to determine alternative representations that might lead to improved performance of the network. Improvement in performance was obtained by modifying the number of hidden layers and units and by suitably selecting the training set and training schedule. While these factors influence performance and need to be controlled, representation also has a considerable influence on performance, as demonstrated both in ref. 11 and the present study.

5 DOMAIN KNOWLEDGE AND REPRESENTATION OF PATTERNS

In most structural engineering applications, performance of the network was improved by providing domain knowledge in addition to information about the domain that is implicit in the training data set. The selection of suitable input/output pattern pairs, clustering of data based on features of the domain, and providing bounds for variables are all techniques that make additional domain knowledge available to the network and hence assist the learning process. Domain knowledge also can be made available to the network in the form of information on which of the variables that characterize the system can be grouped together in the mapping function to be learned. Dimensional analysis provides this information by identifying dimensionless representations of variables in terms of which the mapping function can be expressed.

Other representations for the input/output pattern pairs are possible. In the functional-link approach, the initial representation of a pattern is enhanced by describing it in a space of increased dimensions. This results in the input pattern being mapped on to a larger pattern space. This representation, however, does not introduce any new information. This approach has been shown to increase the learning rate and allows the use of a flat net with no hidden layers.[24]

In this section, dimensional analysis and the information it provides are initially considered, and the means of making this information available to the network are then outlined. The representation resulting from dimensional analysis is used to define the space in which learning is to take place. Though an enhanced representation based on the functional link approach can be selected to further improve performance, it has not been considered in the present study.

5.1 Dimensional analysis

Dimensional analysis is a well-established engineering technique that has been used for purposes of modeling and similitude. Recently, the method has been used in artificial intelligence as a tool for problem solving, including qualitative reasoning problems, and discovery.[3,23]

The physical representations of variables have numerical and symbolic components. The symbolic components are

the dimensional representations of the physical variables, and they encode a significant amount of physical knowledge and are subject to a set of rules. Three such rules are dimensional homogeneity, the product rule, and the Buckingham's π theorem.[23] The most widely used rule in dimensional analysis is the Buckingham's π theorem, which in one form can be stated as follows: If a physical situation is characterized by n variables and r basic dimensions occur in the dimensional representations of these variables, then there are $n - r$ independent dimensionless products that are sufficient to describe the situation.[5] These dimensional principles constrain all physical laws and hence are applicable to all physical systems. There are a number of ways in which the dimensionless products can be generated. Heuristics are available for partitioning the variables and constructing these products.[3,5,23]

5.2 Dimensionless forms of representations

In general, it is possible to express the domain relationship in terms of $n - r$ independent dimensionless products. It is possible, in some situations, to combine the dimensionless products and further reduce the number of independent dimensionless products required to describe this relationship. Dimensional analysis thus leads to a reduction in the dimensionality of the space in which the domain relationship is defined, makes the regularities in the domain explicit, and results in simpler relationships. It is also possible to derive sensitivity information from the dimensionless products, and this information has been used for qualitative reasoning tasks.[3,23]

In the present study, the dimensional analysis provides the basis for identifying dimensionless variables, and these are then used to define the representational framework within which the network learns the mapping function. The domain knowledge is thus embedded into the representational framework, and both the numerical and symbolic components of the knowledge in the physical representation of the variables are made available to the network.

6 REPRESENTATION AND PERFORMANCE—APPLICATIONS

The three problems solved in ref. 29 using neural networks are selected for comparison purposes and to study the effect of representation in learning. The back-propagation algorithm has a number of parameters built in that could be used to improve its convergence characteristics. Since all the parameter settings for these three problems were not reported, these problems were solved initially using the same training and testing sets and representations as in ref. 29. The results obtained for all three problems were very similar to those reported and show the same trends, including poor performance for the same patterns. This ensures that all improvements in performance are attributable to the representational change rather than to the parameter settings.

6.1 Load position—Bending moment patterns in beams

In this problem, the network was trained initially using the bending moment pattern in a simply supported beam subject to a concentrated load. The input values were the bending moments at 10 different sections along the span, and the location of the load represents the output value. The network was then tested using 8 bending moment patterns, 4 of which were new. The results are shown in Table 1. The network performed well on the patterns used for training but poorly in some of the test patterns. The percentage error values (ref. 29 values are shown in parenthesis) for the two sets of results are very close. The slightly higher accuracy observed in the present work is due primarily to the parameter settings selected.

Table 1
Results for simple beam network—Representation in ref. 29

Pattern	*Actual output from ref. 29*	*Actual output from present work*	*Desired output*	*Percentage error (%) (ref. 29 values in parentheses)*
1	0.207	0.200	0.200	0.0 (3.5)
2	0.398	0.399	0.400	0.25 (0.5)
3	0.607	0.601	0.600	0.17 (1.2)
4	0.794	0.798	0.800	0.25 (0.8)
5	0.271	0.273	0.300	9.0 (9.7)
6	0.503	0.500	0.500	0.0 (0.6)
7	0.746	0.748	0.700	6.9 (6.6)
8	0.629	0.637	0.900	29.2 (30.1)

Five variables characterize this problem (bending moment M_δ, load P, their respective locations δ and x, and span L). From dimensional analysis, two independent dimensionless variables are sufficient to describe this situation. The dimensionless variables selected are $M_\delta/P\delta$ and x/L.

In ref. 29, the bending moment patterns used for training were for the load placed at points 1/9, 3/9, 5/9, and 7/9 of span from the left end. Though these are the second, fourth, sixth, and eighth points along the span, they have been incorrectly assumed to be 0.2, 0.4, 0.6, and 0.8 units from the left end of a beam of unit length. In order to avoid this error distorting the present investigation, the output locations have been corrected, and the comparison based on the corrected values is given in Table 2. The percentage error values for the representation of ref. 29 are given in parentheses. A three-layer network with two hidden layers, each having 10 units, was used as in ref. 29. The parameter settings for the back-propagation algorithm also were kept the same for both representations considered in Table 2. In this problem, casting the variables in dimensionless form did not result in a reduction in dimensionality of the space in which learning takes place, and yet there is a dramatic improvement in the performance, particularly in the prediction of the network for the test cases. This indicates an improvement in the generalization capabilities of the network, even for test cases which are outside the range of the training set.

6.2 Design of singly reinforced concrete beams

The design problem considered is that of selecting the depth of a singly reinforced rectangular concrete beam to provide the required ultimate moment capacity. Six variables (bending moment M_u, steel and concrete strengths f_y and f_c', reinforcement ratio ρ, width b, and depth d) are required to characterize this design problem. Using dimensional analysis, the number of independent dimensionless variables required to describe the design is three. The following dimensionless variables are selected:

$$\frac{M_u}{f_c' d^3}, \quad \rho \frac{f_y}{f_c'}, \quad \text{and} \quad \frac{b}{d}$$

For this problem, the explicit form of the mathematical model is available[9] and can be expressed in the form

$$\frac{M_u}{f_c' bd^2} = k_1 \rho \frac{f_y}{f_c'} \left(1 - k_2 \rho \frac{f_y}{f_c'} \right)$$

where k_1 and k_2 are functions of f_c' but become constants if a rectangular stress block for concrete is assumed.

Thus only two dimensionless variables

$$\frac{M_u}{f_c' bd^2} \quad \text{and} \quad \rho \frac{f_y}{f_c'}$$

are sufficient to provide an adequate representation. The relationship is quadratic and hence easy to learn. If the representation in ref. 29 is used, the relationship takes the form

$$\frac{M_u}{f_c'(b/d)} \times \frac{1}{k_1 \rho \frac{f_y}{f_c'} \left(1 - k_2 \rho \frac{f_y}{f_c'} \right)} = d^3$$

Clearly, this is a complex function in terms of the independent variables M_u, f_c', f_y, ρ, and (b/d) and the dependent variable d. Learning in a space defined by these six variables is likely to be slow.

A network with five input units, two hidden layers of six units each, and one output unit was used in ref. 29. Twenty-one randomly chosen patterns were used to train the net-

Table 2
Effect of representation on simple beam network

Pattern	*Representation in ref. 29*	*Dimensionless representation*	*Desired output*	*Percentage error (%) (ref. 29 values in parentheses)*
1	0.1115	0.1113	0.1111	0.18 (0.36)
2	0.3325	0.3325	0.3333	0.24 (0.24)
3	0.5566	0.5567	0.5556	0.20 (0.18)
4	0.7768	0.7770	0.7778	0.10 (0.13)
5	0.1921	0.2135	0.2222	3.92 (13.55)
6	0.4442	0.4423	0.4444	0.47 (.05)
7	0.7212	0.6801	0.6667	2.0 (8.17)
8	0.5863	0.8308	0.8889	6.54 (34.04)

work, and 31 patterns, including 10 new ones, were used for testing. In addition to the considerable time taken to train the network, some of the parameter settings had to be manipulated to provide accurate results. Despite the size of the sample and the preceding precautions, one of the test patterns gave an error as high as 40 percent.[29]

This problem was solved initially using the same representation and network topology as in ref. 29. The results obtained were comparable, with the same test patterns giving rise to large errors as reported. A network with one hidden layer with four units was then considered using the representation in ref. 29. This network performed well on the training set but gave much higher percentage errors for the test set, indicating that the second hidden layer improves the generalization capability of the first network.

There could be no direct comparison of performance of networks using dimensionless representation and those using the representation in ref. 29, since there is a large difference in the dimensionality of the space in which learning takes place. From the dimensionless representation of the mathematical model, it is evident that only two dimensionless variables are required for the representation. Hence a simple network with one input and one output unit with one hidden layer having three units was considered.

The results are shown in Table 3. The network was provided with a smaller training set of five cases (patterns 1 to 5 in Table 3), yet the maximum error in prediction was 13.33 percent for all the 31 patterns presented to the network. This error could be further reduced if the training set is better selected. By replacing pattern 5 with pattern 13 in

Table 3
Concrete beam design results—Representation using dimensionless parameters

Pattern	$\frac{M_u}{f_c' bd^2}$	$\rho \frac{f_y}{f_c'}$ *(desired)*	$\rho \frac{f_y}{f_c'}$ *(actual)*	*Percentage error (alternative training set results in parentheses)*
1	0.1944	0.2533	0.2530	0.12 (1.54)
2	0.1585	0.2000	0.2021	1.05 (0.20)
3	0.1307	0.1600	0.1620	1.25 (1.44)
4	0.1138	0.1387	0.1373	1.01 (0.36)
5	0.1882	0.2453	0.2443	0.41 (1.79)
6	0.2157	0.2880	0.2826	1.88 (3.40)
7	0.1003	0.1200	0.1175	2.08 (0.58)
8	0.1715	0.2200	0.2207	0.32 (0.82)
9	0.1980	0.2600	0.2580	0.77 (2.19)
10	0.0940	0.1120	0.1083	3.30 (0.09)
11	0.1307	0.1600	0.1620	1.25 (1.44)
12	0.1745	0.2240	0.2250	0.45 (0.76)
13	0.0646	0.0750	0.0650	13.33 (3.60)
14	0.1115	0.1350	0.1340	0.74 (0.74)
15	0.1860	0.2400	0.2412	0.50 (0.88)
16	0.0818	0.0960	0.0903	5.94 (0.52)
17	0.1176	0.1440	0.1429	0.76 (0.21)
18	0.1631	0.2040	0.2087	2.30 (1.32)
19	0.0687	0.0800	0.0710	11.25 (2.75)
20	0.1158	0.1400	0.1402	0.14 (1.36)
21	0.1596	0.2000	0.2037	1.85 (0.95)
22	0.1761	0.2267	0.2272	0.22 (1.01)
23	0.1673	0.2133	0.2147	0.66 (0.42)
24	0.1596	0.2000	0.2037	1.85 (0.95)
25	0.1894	0.2400	0.2460	2.50 (1.08)
26	0.1660	0.2100	0.2129	1.38 (0.29)
27	0.1770	0.2280	0.2285	0.22 (1.01)
28	0.0922	0.1100	0.1056	4.0 (0.27)
29	0.1332	0.1650	0.1656	0.36 (0.42)
30	0.0910	0.1080	0.1039	3.80 (0.0)
31	0.1445	0.1800	0.1820	1.11 (0.67)

the training set, the maximum error in prediction reduces to 3.6 percent for the 31 patterns. The percentage error in this case is as shown in parentheses in Table 3.

6.3 Maximum bending moments in rectangular plates—Magnitudes and locations

In this problem, the network is trained to predict the magnitudes and locations of maximum bending moments in a simply supported rectangular plate subjected to a unit concentrated load.

The input quantities selected in ref. 29 are the dimensions of the plate and the location of the load. Since the load was assumed to be of unit value, these four quantities define the input vector for the network. The output quantities selected were the maximum bending moments in the x and y directions and their respective locations. The output vectors are thus defined by these six quantities. The training and testing patterns were generated using a finite-element analysis program. A network consisting of four input units, two hidden layers with six units each, and an output layer with six units was selected. The network was trained on 30 patterns and was tested using 12 patterns. There is a good correlation between the desired and actual outputs in most cases, but the results of some test cases show significant differences.

The variables characterizing the plate-bending problem are the maximum bending moment (M_x or M_y), load P,

Table 4
Simply supported plate results—Representation as in ref. 29 (training patterns)

					x Bending						*y Bending*					
	Plate		*Load*		*Desired*			*Actual*			*Desired*			*Actual*		
Pattern	x	y	x	y	M	x	y	M	x	y	M	x	y	M	x	y
1	1.00	1.00	0.50	0.50	0.32	0.50	0.50	0.34	0.50	0.51	0.32	0.50	0.50	0.33	0.50	0.51
2	1.00	1.00	0.60	0.15	0.24	0.57	0.17	0.25	0.56	0.18	0.26	0.57	0.17	0.26	0.56	0.18
3	1.00	1.00	0.80	0.25	0.27	0.77	0.28	0.28	0.76	0.29	0.26	0.77	0.28	0.27	0.76	0.29
4	1.00	1.00	0.20	0.75	0.27	0.22	0.73	0.28	0.23	0.73	0.26	0.22	0.73	0.26	0.23	0.73
5	1.00	1.00	0.20	0.20	0.25	0.22	0.22	0.26	0.21	0.22	0.25	0.22	0.22	0.25	0.21	0.22
6	0.90	1.00	0.45	0.50	0.33	0.45	0.50	0.34	0.45	0.50	0.32	0.45	0.50	0.32	0.45	0.50
7	0.90	1.00	0.14	0.60	0.26	0.16	0.57	0.27	0.16	0.57	0.23	0.16	0.57	0.24	0.16	0.58
8	0.90	1.00	0.45	0.75	0.29	0.45	0.73	0.29	0.45	0.73	0.30	0.45	0.73	0.31	0.45	0.74
9	0.90	1.00	0.68	0.20	0.26	0.65	0.22	0.27	0.66	0.23	0.26	0.65	0.22	0.27	0.66	0.23
10	0.90	1.00	0.72	0.90	0.20	0.70	0.88	0.21	0.70	0.88	0.21	0.70	0.88	0.21	0.70	0.88
11	1.00	0.80	0.50	0.40	0.31	0.50	0.40	0.31	0.50	0.40	0.33	0.50	0.40	0.33	0.50	0.40
12	1.00	0.80	0.60	0.12	0.22	0.57	0.14	0.22	0.58	0.14	0.26	0.57	0.14	0.26	0.58	0.14
13	1.00	0.80	0.90	0.64	0.21	0.88	0.62	0.22	0.88	0.62	0.20	0.88	0.62	0.20	0.88	0.62
14	1.00	0.80	0.20	0.60	0.26	0.22	0.58	0.27	0.23	0.58	0.27	0.22	0.58	0.28	0.23	0.58
15	1.00	0.80	0.75	0.40	0.30	0.73	0.40	0.30	0.73	0.40	0.30	0.73	0.40	0.31	0.73	0.40
16	0.70	1.00	0.14	0.20	0.26	0.11	0.22	0.26	0.12	0.23	0.23	0.11	0.22	0.23	0.12	0.23
17	0.70	1.00	0.10	0.60	0.25	0.12	0.57	0.25	0.12	0.58	0.21	0.12	0.57	0.21	0.12	0.58
18	0.70	1.00	0.17	0.80	0.27	0.19	0.77	0.27	0.19	0.77	0.25	0.19	0.77	0.25	0.19	0.78
19	0.70	1.00	0.52	0.20	0.27	0.51	0.22	0.27	0.51	0.22	0.25	0.51	0.22	0.25	0.51	0.22
20	0.70	1.00	0.35	0.75	0.30	0.35	0.73	0.30	0.35	0.73	0.29	0.35	0.73	0.30	0.35	0.74
21	1.00	0.60	0.20	0.12	0.22	0.22	0.14	0.22	0.22	0.15	0.25	0.22	0.14	0.25	0.22	0.15
22	0.60	1.00	0.09	0.60	0.24	0.10	0.57	0.24	0.11	0.58	0.20	0.10	0.60	0.20	0.11	0.59
23	1.00	0.60	0.80	0.15	0.24	0.77	0.17	0.23	0.77	0.17	0.27	0.77	0.17	0.26	0.77	0.17
24	0.60	1.00	0.45	0.20	0.27	0.44	0.22	0.27	0.44	0.22	0.24	0.44	0.22	0.24	0.44	0.22
25	0.60	1.00	0.30	0.50	0.32	0.30	0.50	0.31	0.30	0.50	0.28	0.30	0.50	0.27	0.30	0.50
26	1.00	0.50	0.50	0.25	0.26	0.50	0.25	0.26	0.50	0.25	0.32	0.50	0.25	0.31	0.50	0.25
27	0.50	1.00	0.13	0.80	0.26	0.14	0.77	0.26	0.14	0.77	0.23	0.14	0.77	0.23	0.14	0.77
28	1.00	0.50	0.75	0.25	0.26	0.73	0.25	0.26	0.73	0.25	0.30	0.73	0.25	0.30	0.73	0.25
29	0.50	1.00	0.40	0.90	0.23	0.22	0.36	0.23	0.22	0.36	0.26	0.22	0.36	0.26	0.22	0.36
30	1.00	0.50	0.60	0.08	0.18	0.60	0.09	0.19	0.60	0.09	0.23	0.60	0.09	0.24	0.60	0.09

Table 5
Simply supported plate results—Representation as in ref. 29 (test patterns)

					x Bending						*y Bending*					
	Plate		*Load*		*Desired*			*Actual*			*Desired*			*Actual*		
Pattern	*x*	*y*	*x*	*y*	*M*	*x*	*y*	*M*	*x*	*y*	*M*	*x*	*y*	*M*	*x*	*y*
31	1.00	1.00	0.75	0.50	0.30	0.73	0.50	0.32	0.74	0.52	0.29	0.73	0.50	0.30	0.74	0.52
32	1.00	1.00	0.90	0.80	0.21	0.88	0.77	0.17	0.91	0.83	0.19	0.88	0.77	0.12	0.91	0.83
33	0.90	1.00	0.18	0.20	0.25	0.20	0.22	0.26	0.18	0.22	0.25	0.20	0.22	0.25	0.18	0.22
34	0.90	1.00	0.22	0.80	0.26	0.25	0.73	0.27	0.24	0.77	0.26	0.25	0.73	0.26	0.24	0.77
35	1.00	0.80	0.60	0.12	0.22	0.57	0.14	0.22	0.58	0.14	0.26	0.57	0.14	0.26	0.58	0.14
36	1.00	0.80	0.80	0.20	0.26	0.77	0.22	0.25	0.76	0.22	0.27	0.77	0.22	0.27	0.76	0.22
37	0.70	1.00	0.35	0.50	0.33	0.35	0.50	0.32	0.35	0.49	0.30	0.35	0.50	0.30	0.35	0.50
38	0.70	1.00	0.56	0.90	0.21	0.54	0.88	0.25	0.46	0.74	0.21	0.54	0.88	0.27	0.46	0.74
39	0.60	1.00	0.30	0.75	0.30	0.30	0.73	0.30	0.30	0.74	0.28	0.30	0.73	0.28	0.30	0.74
40	1.00	0.60	0.90	0.48	0.20	0.88	0.47	0.25	0.85	0.41	0.21	0.88	0.47	0.26	0.85	0.41
41	0.50	1.00	0.10	0.20	0.25	0.11	0.22	0.25	0.07	0.23	0.21	0.11	0.22	0.22	0.07	0.23
42	1.00	0.50	0.20	0.38	0.23	0.22	0.36	0.24	0.18	0.32	0.26	0.22	0.36	0.27	0.18	0.32

location of load (x_1, y_1), and the plate dimensions (a, b). The dimensionless variables selected are M_x/P (or M_y/P), x_1/a, y_1/b, and b/a. The location (x_2, y_2) of the maximum bending moment section can be considered as a response quantity, and a similar set of dimensionless variables can then be selected for the location problem from dimensional analysis. The decomposition of the problem posed into four simpler problems permits the network to learn the four separate functions that define the variations of the four response quantities.

As in the preceding cases, the problem is solved initially using the representation in ref. 29, and results similar to those reported were obtained and are shown in Tables 4 and 5. The network was then trained using the dimensionless parameters $b/2a$, x_1/a, and y_1/b providing the input values and M_x/P providing the output value. By partitioning the input and output patterns in this form, the relationships that exist between the different sets of parameters were uncoupled, thereby reducing the dimensionality of the space to be mapped. A separate network was then trained with M_y/P as the output value and the same parameters providing the input patterns as for the first network.

The results obtained by the two networks for moments are shown in Tables 6 and 7, where they are also compared with the results in ref. 29. Significant improvement in performance can be seen mainly for the test data set, indicating that the dimensionless representation of variables results in improved generalization capability. Similar improvements are also observed in the results for the location of the maximum moment sections. Pattern 29 was left out of the training set, since the location of the maximum moment section appears to be incorrect.

7 CONCLUSIONS

The performance of neural networks in structural engineering applications can be improved significantly by selecting a suitable representational framework in which to present the training input/output pattern pairs. Dimensional analysis provides a means of selecting such representations and permits the embedding of domain knowledge into the representation and reducing the dimensionality of the space in which the mapping function is to be learned. In the three cases considered, dimensionless representations resulted in improved performance, mainly in the generalization aspects of learning, and this was achieved without introducing any of the available domain-independent techniques. As demonstrated in the reinforced concrete beam design problem, a dimensionless representation results in a simpler mapping function and makes it possible to train the network on a smaller data set and still have the capability for reasonably accurate predictions.

The real usefulness of neural networks in structural engineering is not in replacing existing algorithmic approaches for predicting structural response, as a computationally efficient alternative, but in providing concise relationships that capture previous design and analysis experiences that are useful for making design decisions. The representations that lead to efficient learning in neural networks also define the spaces in which the search for such high-level relationships for structural design should be conducted.

Though the present work has been developed in the context of structural engineering, the approach is fairly general and can be easily applied to other domains where dimensional analysis is applicable.

Table 6
Simply supported plate results—Effect of representation (training patterns)

Pattern	M_x *(desired)*	M_x *(actual, ref. 29)*	M_x *(actual)*	M_y *(desired)*	M_y *(actual, ref. 29)*	M_y *(actual)*
1	0.32	0.32	0.32	0.32	0.33	0.32
2	0.24	0.24	0.24	0.26	0.27	0.26
3	0.27	0.25	0.27	0.26	0.26	0.27
4	0.27	0.27	0.27	0.26	0.25	0.26
5	0.25	0.25	0.25	0.25	0.27	0.25
6	0.33	0.32	0.33	0.32	0.32	0.31
7	0.26	0.25	0.26	0.23	0.23	0.24
8	0.29	0.31	0.29	0.30	0.31	0.30
9	0.26	0.26	0.26	0.26	0.27	0.26
10	0.20	0.20	0.20	0.21	0.21	0.21
11	0.31	0.31	0.31	0.33	0.34	0.33
12	0.22	0.22	0.22	0.26	0.26	0.26
13	0.21	0.21	0.21	0.20	0.22	0.20
14	0.26	0.26	0.26	0.27	0.26	0.27
15	0.30	0.29	0.30	0.30	0.30	0.30
16	0.26	0.24	0.26	0.23	0.23	0.23
17	0.25	0.25	0.25	0.21	0.22	0.21
18	0.27	0.28	0.27	0.25	0.24	0.25
19	0.27	0.26	0.27	0.25	0.25	0.25
20	0.30	0.30	0.30	0.29	0.28	0.30
21	0.22	0.22	0.22	0.25	0.25	0.25
22	0.24	0.25	0.24	0.20	0.21	0.20
23	0.24	0.25	0.24	0.27	0.26	0.26
24	0.27	0.27	0.27	0.24	0.25	0.24
25	0.32	0.31	0.32	0.28	0.29	0.28
26	0.26	0.26	0.26	0.32	0.30	0.33
27	0.26	0.27	0.26	0.23	0.21	0.23
28	0.26	0.28	0.26	0.30	0.29	0.29
29	0.23	0.24	0.23	0.26	0.25	0.26
30	0.18	0.19	0.18	0.23	0.23	0.23

Table 7
Simply supported plate results—Effect of representation (test patterns)

Pattern	M_x *(desired)*	M_x *(actual, ref. 29)*	M_x *(actual)*	M_y *(desired)*	M_y *(actual, ref. 29)*	M_y *(actual)*
31	0.30	0.24	0.32	0.29	0.25	0.30
32	0.21	0.15	0.21	0.19	0.15	0.23
33	0.25	0.25	0.26	0.25	0.26	0.25
34	0.26	0.28	0.26	0.26	0.26	0.26
35	0.22	0.22	0.22	0.26	0.26	0.26
36	0.26	0.26	0.26	0.27	0.27	0.27
37	0.33	0.32	0.33	0.30	0.31	0.29
38	0.21	0.22	0.21	0.21	0.25	0.24
39	0.30	0.29	0.31	0.28	0.26	0.29
40	0.20	0.27	0.21	0.21	0.28	0.17
41	0.25	0.25	0.24	0.21	0.23	0.21
42	0.23	0.20	0.24	0.26	0.23	0.27

ACKNOWLEDGMENTS

This work has received limited financial support from the University Research Grant Scheme of the University of Sydney. The authors are grateful to Dr. Marwan Jabri, Department of Electrical Engineering, University of Sydney, for making the neural network program DIME available for this study.

REFERENCES

1. Adeli, H. & Yeh, C., Perceptron learning in engineering design. *Microcomp. Civil Eng.* 4(4) (1989), 247–56.
2. Berke, L. & Hajela, P., Application of neural nets in structural optimization. *AGARD ASI*, Berchtesgaden, 1991.
3. Bhaskar, R. & Nigam, A., Qualitative physics using dimensional analysis. *Artificial Intelligence* **45**(1–2) (1990), 73–111.
4. Brown, D. A., Murthy, P. L. N. & Berke, L., Computational simulation of composite ply micromechanics using artificial neural networks. *Microcomp. Civil Eng.* **6** (1991), 87–97.
5. Cowan, H. J., Gero, J. S., Ding, D. & Muncey, R. W., *Models in Architecture*. Elsevier, Amsterdam, 1968.
6. Fahlman, S. C., Faster-learning variations on back-propagation: An empirical study. In *Proceedings of the 1988 Connectionist Models Summer School*, Morgan Kaufmann, San Mateo, Calif., 1988, 38–51.
7. Gero, J. S., The design of cable network structures. In *Proceedings of the Second Conference on Steel Developments*, Melbourne, Australia, 1974, 129–34.
8. Ghaboussi, J., Garrett, J. H. & Wu, X., Knowledge-based modeling of material behaviour with neural networks. *J. Eng. Mech.* **117**(1) (1991), 132–53.
9. Gunaratnam, D. J., Integrated design charts for reinforced concrete flexural sections. *Proc. Inst. Civil Eng.*, **83**(2) (1987), 443–51.
10. Gunaratnam, D. J. & Bhattacharya, A. P., Transverse vibrations and stability of polar orthotropic circular plates: High-level relationships. *J. Sound Vibration* **137**(3) (1989), 383–92.
11. Gunaratnam, D. J. & Gero, J. S., High-level relationships for structural design: The effect of representation in learning. In *The Technology of Design*, ed. G. Woodbury, ANZAScA, University of Adelaide, Adelaide, 1991, 127–34.
12. Hajela, P. & Berke, L., Neurobiological computational models in structural analysis and design. *Comput. Struct.* **41**(4) (1991), 657–67.
13. Hajela, P., Fu, B. & Berke, L., ART networks in automated conceptual design of structural systems. In *Artificial Intelligence and Structural Engineering*, ed. B. H. V. Topping, Civil-Comp Press, Edinburgh, 1991, 263–71.
14. Hajela, P., Fu, B. & Berke, L., Self-organization in neural networks applications: In structural optimization, *AGARD ASI*, Berchtesgaden, 1991.
15. Hajela, P. & Berke, L., Neural networks in structural analysis and design: An overview. *Comput. Syst. Eng.* **3**(1–4) (1992), 525–38.
16. Hertz, J., Krogh, A., & Palmer, R. G., *Introduction to the Theory of Neural Computation*, Addison-Wesley, Reading, Mass., 1991.
17. Hinton, G. E., Connectionist learning procedures. In *Machine Learning: Paradigms and Methods*, ed. J. G. Carbonell, MIT Press, Cambridge Mass., 1990, 185–234.
18. Hung, S. L. & Adeli, H., A model of perceptron learning with a hidden layer for engineering design. *Neurocomputing* **3** (1991), 3-14.
19. Hung, S. L. & Adeli, H., Parallel backpropagation learning algorithm on Cray Y-MP8/864 supercomputer. *Neurocomputing* (in press).
20. Lippmann, R. P., An introduction to computing with neural nets. In *Artificial Neural Networks*, ed. V. Vemuri, IEEE, 1988, 36–54.
21. Liu, X. & Gan, M., A preliminary design expert system (SPRED-1) based on neural network. In *Artificial Intelligence in Design '91*, ed. J. S. Gero, Butterworth-Heineman, Oxford, 1991, 785–800.
22. Maren, J. A., Harston, C. T. & Pap, R. M., *Handbook of Neural Computing Applications*, Academic Press, San Diego, Calif., 1990.
23. Nigam, A. & Bhaskar, R., Qualitative reasoning about engineering systems using dimensional analysis. In *Artificial Intelligence in Design*, ed. J. S. Gero, Springer-Verlag, Berlin, 1989, 273–90.
24. Pao, Y. H., *Adaptive Pattern Recognition and Neural Networks*, Addison-Wesley, Reading, Mass., 1989, 197–221.
25. Pao, Y. H., Knowledge in the form of patterns and neural network computing. In *Knowledge Engineering*, Vol. 1: *Fundamentals*. ed. H. Adeli, McGraw-Hill, New York, 1990, 200–25.
26. Ramirez, M. R. & Arghya, D., A faster learning algorithm for back-propagation neural networks in NDE applications. In *Artificial Intelligence and Civil Engineering*, ed. B. H. V. Topping, Topping, Edinburgh, 1991, 275–83.
27. Rich, E. & Knight, K., *Artificial Intelligence*, McGraw-Hill, New York, 1991, 487–528.
28. Rubinstein, M. F., *Tools for Thinking and Problem Solving*, Prentice-Hall, Englewood Cliffs, N.J., 1986.
29. Vanluchene, D. & Roufei, S., Neural networks in structural engineering. *Microcomp. Civil Eng.* **5** (1990), 207–15.