NASA Technical Memorandum 102420

Applications of Artificial Neural Nets in Structural Mechanics

Laszlo Berke
Lewis Research Center
Cleveland, Ohio

and

Prabhat Hajela
Rensselaer Polytechnic Institute
Troy, New York

Presented at the
Lecture Series on "Shape and Layout Optimization of Structural Systems"
at the International Centre for Mechanical Sciences
Udine, Italy, July 16–20, 1990

APPLICATION OF ARTIFICIAL NEURAL NETS

IN STRUCTURAL MECHANICS*

Laszlo Berke
National Aeronautics and Space Administration
Lewis Research Center
Cleveland, Ohio 44135

and

Prabhat Hajela†
Rensselaer Polytechnic Institute
Troy, New York 12181

ABSTRACT

A brief introduction to the fundamentals of Neural Nets is given first, followed by two applications in structural optimization. In the first case the feasibility of simulating with neural nets the many structural analyses performed during optimization iterations was studied. In the second case the concept of using neural nets to capture design expertise was investigated.

1. INTRODUCTION

Considerable activity can be observed in the development and application of a certain class of trainable network paradigms, namely the biologically motivated Artificial Neural Nets (ANN). This upsurge of developmental activities is expected to contribute to the availability of powerful new capabilities in the near future. It is this expectation that motivated the examination of the usefulness of ANN in structural optimization as one of the many potential applications in structural analysis and design.

It appears that only a few structural problems were investigated as demonstrations of neural net capabilities. References 1 and 2 are examples dealing with simple oscillators and a beam design problem, respectively. As will be discussed in subsequent sections, there are many potentially productive applications.

*Presented at the Lecture Series on "Shape and Layout Optimization of Structural Systems" at the International Centre for Mechanical Sciences, Udine, Italy, July 16-20, 1990.

†Summer Faculty Fellow at NASA Lewis Research Center.

Developments in Computational Structures Technology (CST) are closely linked to developments in computational capabilities. ANN technology, and its exploitation for CST falls in the same category and an initial investigation, as reported here, appeared warranted. CST is very demanding on computer resources and new approaches for their utilization are continuously being explored. For example it is not widely known that the first engineering application of Artificial Intelligence (AI) in the form of a rule based expert systems was in structural analysis. This first expert system capability, Structural Analysis Consultant (SACON) became somewhat of a classic model for similar applications (refs. 3,4). AI is now a widely accepted technology for engineering applications as illustrated by the large number of conferences and publications.

Artificial neural nets are also a class of AI paradigms, and provide new opportunities for applications of computer science developments in CST. This brief note proposes a few potentially profitable applications and presents results of feasibility studies associated with automated structural design.

Only the basic ideas of artificial neural nets will be provided in this brief study. References 5 and 6 are recommended for those interested in a detailed introduction to neural nets. Reference 6 also describes an extension of neural nets called "functional links." This represents an extension of the biologically motivated approaches towards more specific mathematical functionalities present in engineering or other applications.

The feasibility studies presented here were conducted using well known small "toy" problems. Research is underway to explore the limits of applicability. This includes increasing problem complexity and dimensionality as indicated by the number of input-output variables, and the nature of nonlinearities in their functional dependence. As will be discussed next the "training" of neural nets involve the minimization of some error measure. Consequently, limitations on dimensionality of problems associated with optimization techniques also apply, at least in the case of ANN utilized in this study. ANN hardware is the subject of vigorous developments and eventually may provide considerable increase in processing power.

2. BASIC CONCEPTS OF ARTIFICIAL NEURAL NETS

As stated earlier, only a brief and incomplete introduction is given here, and References 5 and 6 are suggested as introductory reading. Many other texts are available, and the body of publications is increasing very rapidly.

Neural nets can be viewed in different ways. The original motivation came from creating computer models that can mimic certain brain functions, and the word neural was attached to the designation of this class of models. For computer specialists ANN are a class of parallel distributed processors with some particular processing capability in the artificial neurons and

modification of data during communication among them. The particular class of ANN utilized in this study can be viewed either as brain function models if one is romantically inclined, or simply as a flexible technique for creating nonlinear regression models for data with complex interdependencies.

Figure 1a shows a simplistic representation of a neuron with the following components of interest: a cell body with the mechanism which controls cell activity, the "axon" that transmits stimulus from one neuron to others, the "dendrites" which also receive electrical signals from connected neurons or from an external source, and the "synapses" which define interconnections and their respective strengths. In a human brain the number of neurons approaches a trillion, each connected perhaps to tens of thousands of other neurons forming an immense network. Figure 1b shows a small segment of this network in the cerebral cortex.

Artificial neural nets were conceived as very simple models of certain brain activities. Of interest for us here are those aspects of biological neural net activities that are associated with learning, memory, and generalization from accumulated experience. Learned information is thought to be represented by a pattern of synaptic connection strengths that modify the incoming stimuli, strengthening or inhibiting them. When the accumulation of the received stimuli in the neuron reaches a certain threshold, it "fires," sending out an electrical stimulus to all connected neurons. Learning in turn is thought to be associated with the development and retention of a pattern of the connection strengths in various regions of this immense network. It has been suggested that such retained patterns are somewhat similar in nature to holograms that also contain complex information in a vast arrangement of simple patterns.

Artificial neural nets simulate the above activities in brain tissue through very simple concepts. An artificial neuron receives information labeled x_i from the incoming n connections from other neurons as indicated in Figure 2a. Such neurons and their connections can be assembled in principle into any architecture of connectivities as indicated in Figure 2b. The information x_i sent out by the connecting neurons and received by the jth neuron of a net are modified by connection strengths w_{ij}. The jth neuron performs a summation of the modified information as also indicated in Figure 2a., and processes the value r of the sum through an activation function producing an output z_j. This output is then sent as a stimulus to all connecting neurons, and determines in turn, the activity of those neurons. Figure 3 shows a few activation functions, with the sigmoid function being the most popular. More complex neuron activation functions can be devised for various special purposes.

Training of neural nets of interest here involves the evolution of the connection strengths w_{ij} everywhere in the net through "training". Sets of known input and associated output values are presented to the net and the w_{ij} are adjusted during an iterative procedure to minimize a selected error measure between the desired output and the one produced by the net. Once

trained, the network responds to a new input within the domain of its training by "propagating" it through the net and producing an output. This output is an estimate within certain error, of the output that the actual computational or physical process would have produced.

Several neural net paradigms have emerged as a result of over four decades of research, each with its own purpose and capabilities. The particular class of neural nets that are of interest to us here fall in the category of "feed forward" nets because the input data given to the network is propagated forward towards the output nodes. The "delta-error back propagation" algorithm (see Refs. 5 or 6) is used usually for their "supervised" learning. It is essentially a special purpose steepest descent algorithm to adjust the w_{ij} connection strengths, and other additional internal parameters that are sometimes added to increase flexibility. In principle other optimization methods can also be used, and the development of efficient learning algorithms is an active area of research.

Most currently available neural net capabilities are simulations of the distributed parallel processing concept on serial machines, and such simulations were also used in this study. Neural nets present premier applications for parallel machines or for the developments of special purpose hardware. These approaches are all being investigated, and neural nets enjoy vigorous funding and developments worldwide. As mentioned before, it is this fact that served as motivation for the present study. Other CST applications are also being investigated in view of expected increases in capabilities.

To start out with an application one requires a set of known input and output pairs that must be generated by the "real" process one is planning to simulate. The number of training pairs, and how they span the intended domain of training, is part of what is still an art in ANN requiring experimentation and experience. The same statement is also valid for the architecture of the neural net one intends to use. The examples given later will provide some idea of what is required for a successful application. For the engineering applications presented here, it is perhaps worthwhile to think of neural nets as a peculiar automated multidimensional surface fitting or nonlinear regression capability. What one would accept as a representative input-output set to produce a useful surface fit, is most likely a good start to determine the training pairs for the neural nets.

For the present application it is sufficient to discuss the simplest forms of net architectures. A single layer net is called a "flat" net and is of little interest here in its basic form. It has limited capabilities to represent nonlinearities unless these are specifically captured in the input. An example of this is the use of reciprocal variables in problems involving structural stiffness, a case to be discussed later. Reference 6 provides a powerful generalization of this concept referred to as "functional links". In general, nets have an input and output layer with the number of neurons in each of these matching the number of input and output variables, respectively, and one or more "hidden layers." As an

example, Figure 4 has two nodes in its input layer, three nodes in its single hidden layer, and one node in its output layer. In later discussions such a net will be designated a (2,3,1) net signifying the number of nodes in its three layers. A net can provide an n-to-m mapping, which, for the case of Figure 4, is a 2-to-1 mapping.

The number n and m of nodes in the input and output layer is determined by the number of input and output variables in the training set. It is however, important to determine the necessity of one or more "hidden" layers in the network. A single hidden layer with nodes numbering somewhere between the average and the sum of the input and output nodes is suggested in the literature as a good first start. To add more layers for added flexibility is a temptation which must be resisted in the simple cases addressed in this note. A general suggestion is to try to use as few nodes as possible. As in any optimization problem one should avoid needless increase in the number of optimization variables.

The n-to-m mappings discussed here can also be separated into m n-to-1 mappings or a number of mappings involving groups of output variables. Very large problems may have to be separated that way to keep the number of learning parameters within a practical range for any of the single mappings. For the small problems discussed here, and based on limited experimentation, the training times for one n-to-m mapping appeared to be slightly more favorable than for the equivalent m n-to-1 mappings to equal accuracy.

Once an architecture has been selected, the training starts out with a random set of connection weights w_{ij} usually generated automatically by the particular capability used. These connection weights are then adjusted by a learning algorithm to minimize the difference between the training output values and the values produced by "propagating" the associated input through the net. The training is sensitive to the choices of the various net learning parameters. The principal parameters are the "learning rate" which essentially governs the "step size," a concept familiar to the optimization community, and the "momentum coefficient" which forces the search to continue in the same direction to aid numerical stability, and to go over local minima encountered in the search.

During supervised learning these parameters are adjusted periodically based on the changing convergence trend during iterations. In the "Ten Bar Truss Optimum Design Expert" example discussed later, a publicly available NASA developed capability NETS 2.0 was used. Its user manual, Reference 7, provides a good introduction for someone who would like to experiment with neural nets. NETS 2.0 has a number of other learning parameters and provides good default values for them, including some adaptive features during training iterations. A few possible applications within CST are suggested next, followed by a representative set of the results obtained in preliminary feasibility studies.

3. NEURAL NETS IN COMPUTATIONAL STRUCTURES TECHNOLOGY

The history of the exploitation of computer technology by CST can be viewed, even if somewhat romantically, as attempted simulations of the brain processes of an expert designer at higher and higher levels of abstraction. Procedural codes, expert systems, and neural nets represent this higher and higher levels of abstraction from "number crunching," "expert judgments" and finally a "feel" for a problem area, respectively. These three levels represent increasing intellectual levels and ability to provide quick expert estimates for solutions with less and less participation required of the human user. The final aspiration of researchers in CST is the development of automated expert design capability; neural nets perhaps provide an approach towards that goal.

As described earlier, artificial neural nets perform their functions by developing specific "patterns" of their connection weights. These patterns, and not any individual value serves as the storage of the knowledge. It would be naive to make much of this supposed similarity to brain functions. Much has been learned about the electrochemical activities of brain cells and of the vast neural nets they form. What all that means is poorly understood if at all, and the functioning of the brain remains largely unknown. A more realistic view of the class of neural nets employed in the present study would be that they are essentially glorified surface fitting capabilities. The important consideration is that neural net research activities are expected to result in major novel hardware and software capabilities when compared to other mathematical procedures for nonlinear regression.

The major advantage of a trained neural net over the original (computational) process is that results can be produced in a few clock cycles representing orders of magnitude less computational effort than the original process. This processing time, once the net is trained, is also insensitive to the effort it takes to generate an output by the original process. Consequently benefits can be higher for those problem areas that are computationally very intensive, such as optimization, especially in multidisciplinary settings. There is of course a catch, namely that in those cases the generation of sufficient training data is also costlier. Practical applications can be envisioned where a problem is frequently solved within limited variations of the input parameters. Organizations with specific products for slightly changing applications could develop or evolve trained neural nets based on sets of past solutions. New solutions could then be obtained with negligible efforts. Machine components that are of a certain basic configuration slightly changing from application to application could be good practical examples.

The w_{ij} weights, as they develop, may contain information concerning hidden functional relationships between the variables for some of the applications providing "feature extraction" capabilities. A version of neural nets designated as "unsupervised learning" has such feature extraction capabilities. Training pairs can be preprocessed to be grouped

into clusters based on similarity of features within a certain selected radius. Training effort is then reduced by using the "centers" of such clusters for generalization.

Multidisciplinary design optimization provides particularly intriguing possibilities. For example, nets could be trained for each of the participating disciplines, and integrated to represent appropriate coupling or to use an additional net that develops the important coupling functionalities through feature extraction.

The feasibility of two particular applications at two distinct levels of abstraction were studied in some detail and the results are presented next. The first one involved training a neural net to replace analyses of given structural configurations during optimization iterations. The second exercise was to train a neural net to provide estimates of the actual optimum structures directly totally avoiding the conventional analysis and optimization iterations.

4. NEURAL NET ASSISTED OPTIMIZATION.

This first feasibility study to simulate analysis with the quick response of neural nets was motivated by the approximation concepts in structural optimization. The idea here was to train a neural net to provide computationally inexpensive estimates of analysis output needed for sensitivity evaluations, which in turn is needed by most optimization codes. The numerical experimentation also served to gain initial experience with neural nets.

The familiar five bar and ten bar truss "toy" problems, shown in Figures 5 and 6 respectively, were used for this initial feasibility study. First, various sets of input-output training pairs and network configurations were examined to find the combination that reduced the training effort and produced trained nets which yielded good results as measured by their ability to generalize.

Once an acceptable trained neural net was obtained, it was attached to an optimizer, and all analysis information was obtained from it instead of invoking a conventional analysis capability. Mixing neural net predictions with occasional conventional analyses was not explored, but it is an approach that could possibly exploit the advantages of both.

To create the training sets conventional optimum designs had to be created for two reasons. First, optimum designs were required for comparisons with designs obtained using neural net simulation of the analyses. The second was to perform analyses with random sets of the values of the design variables, in this case the bar areas, within certain preset variations of their optimum values. The optimization involved constraints on the nodal displacements. Consequently, the input-output training pairs for analysis simulation consisted of the bar areas as inputs and nodal

displacements as analysis output variables respectively. How many pairs to use, and within what range of variations, is itself a research question. Because of the nature of the sigmoid function at least the output variables are to be scaled by the user or automatically by the neural net code, to within the most active range of the sigmoid function. Scaling minimum and maximum values to 0.1 and 0.9 is usually suggested.

At this point one has to prescribe the number of iterations for which the network must be trained to obtain desired levels of accuracy. A number from a few hundred to tens of thousands is routinely accepted in neural net applications, even for small nets as in this study. For this level of experimentation one often initiates a run on a PC or a work station and lets it run to a large number overnight in somewhat of an overkill.

Some of the net configurations examined for the five bar truss exercise are shown in Figure 7. As the first attempt a 5-to-4 mapping with a (5,4) net was tried with no hidden layer as shown in Figure 7a. The four output variables were the four nodal displacements indicated in Figure 5. Since the active constraints were essentially related to displacements d_2 and d_4, the rest of the nets considered only these two displacements as output. Figure 7b consequently is a (5,2) net with reduced training effort. Figure 7c is a (10,2) net with the reciprocals of the bar areas also included to help the net capture without a hidden layer the inverse relation between bar areas and nodal displacements. Table 1 contains data on the results of these initial training efforts with other functional relationships also included to try to capture nonlinearities. These attempts without a hidden layer were not totally satisfactory in terms of obtained accuracy or number of required training cycles.

Including a hidden layer, as shown in Figure 7d, produced acceptable results. Table 2 presents the results of optimization using various net and training set combinations. Using the (5,7,2) net and scaled variables, an optimum design was obtained within 2.4% of the exact optimum design proving the feasibility of the basic concept of neural net assisted optimization. Table 3 presents the results of similar experimentation for the ten bar truss supporting the same conclusion. Similar results were obtained for a higher dimensionality wing box problem.

5. NEURAL NETS AS EXPERTS FOR DIRECT OPTIMUM DESIGN ESTIMATES

The next set of experiments were conducted to explore the idea of training a neural net to estimate optimum designs directly for given design conditions and bypass all the analyses and optimization iterations of the conventional approach. It is conceivable in practice that successful similar designs could be collected within some domain of design conditions, input-output pairs defined, and then a neural net trained to serve as "intelligent corporate memory" that can provide a new design for different design requirements instantaneously.

Now let us suppose that we work in a company that markets equipment that is mounted in all cases on ten bar trusses as shown in Figure 6. These trusses have to carry the equipment weight (2 X 100 K) at the two lower free nodes while these support points cannot deflect more than 2 in. The dimensions L_1 and L_2 and H of the trusses can vary between 300 and 400 inches, depending on the particular installation. The engineer who was designing the trusses for the past 30 years and could simply tell the optimum bar areas for any combination of those dimensions has just retired. Can we create an accurate simulation of this departed expert? Yes we can, and rather simply!

To experiment with various training sets, optimum designs were generated by conventional methods for varying first only H in 5 inch increments. The results are given in Table 4. There are three kinds of output numbers in the set. These are the bar areas that change, areas that are at the preselected minimum value of 0.1 for all designs, and the weight, which is of a different order of magnitude. A representative A_1 and A_2, and the optimum weight Wt were considered in the first numerical experiments. The neural net code NETS 2.0 (Ref. 7) was used for all of the direct optimum ten bar truss design exercises.

A number of small net configurations were tried for these 1-to-3 mappings more or less as a learning exercise . Table 5 shows the results of training with a (1,6,3) net, a probable overkill with too many nodes in the hidden layer. Table 6 gives the results of design estimates of the trained net for the remaining check cases of Table 4 that were not included in the training set. The training was performed to 1% RMS accuracy within 200 iterations. As can be seen, both the training accuracy and the estimates for the new cases is around a third of one percent for the individual values and can be considered quite satisfactory. It is also of some interest to note that the net had to evolve its w_{ij} connection weights and other internal parameters provided by NETS 2.0 in such a manner that it could also reproduce a constant .1 value for any input while also producing accurate values of variables of different orders of magnitude for the same inputs.

After the above limited exercises the 3-to-11 mappings were performed between L_1, L_2, H, and the ten bar areas $A_1,\ldots,A_{10}$ and the optimum weight Wt. A rather limited training set of only ten input-output patterns was created using ten random sets of L_1, L_2 and H and the corresponding ten optimum designs. It is interesting to note that ten training pairs did quite well in this case versus the hundreds of training pairs used for the neural net assisted optimization study. Of course, in this problem only three variables are varied to cover a domain. The net used for this exercise is shown in Fig. 8. The ten optimum designs used for training the net are given in Table 7. Optimum designs were then obtained for another seven random sets of L_1, L_2, and H, as checks on the estimates obtained from the trained network. Table 8 shows the seven design conditions and the optimum designs.

The (3,14,11) net given in Figure 8 was used with the nodes in the hidden layer taken as a sum of the input and output nodes. The training was repeated with a (3,6,11) net which also was successful but the learning parameters had to be adjusted after about 100 iterations. During experimentation with various options during training, it was found that it is beneficial to code the 0.1 minimum sizes as 0.5. The active midpoint of the sigmoid activation function is the explanation. This value also represented net accuracy in better detail. NETS 2.0 worked very well, and 1% RMS accuracy was obtainable with 200 iterations in around 30 seconds on a SUN 386i, and using only the default values for the learning parameters for the (3,14,11) net. Because of this good performance, exercises were conducted to overtrain the net. Letting it run for 5000 iterations an RMS accuracy of 0.006% was obtained. Over-training is to be avoided because the neural net at that point becomes a memory with lessened ability to generalize. The overtrained net actually reproduced the training results exactly, but it did a little worse if anything against the seven check conditions than the net trained only for 1% RMS accuracy. The results are shown for one of the check cases in Fig. 9 where the first bar is the desired result, the second is obtained with 1% RMS accuracy and the third is obtained with 0.006% RMS accuracy of training. As can be seen nothing has been gained in accuracy of the estimated results.

Table 9 shows comparisons and the percent error of net estimates for the seven check cases of Table 8 for the net trained for 1% accuracy. As can be seen, the results are quite satisfactory and certainly would be good enough information to produce the ten bar trusses to support the equipment at minimum weight and 2.0 inches maximum deflections. The net produced its estimates by computing a few sums of products in practically no computer time. The mental activities our retired expert designer employed to come up with his optimum designs have been replaced by a trained (3,14,11) neural net of similar capability for this limited task.

6. CLOSING REMARKS

It has been shown that artificial neural nets have intriguing applications in Computational Structural Technology. What has been presented here are two of the possibilities. There are now efforts underway to explore multidisciplinary design applications, to "package" composite material property generation codes as quick response neural net simulations, to develop structural component life prediction capabilities and to capture constitutive material relationships both from theoretical codes and directly from test data. Integrating neural net capabilities with expert systems and optimization algorithms into an automated capability to generate trained neural nets once the domain of interest is defined for often occurring structural components in a design office is also being explored. There are many other applications possible. Controls is one of the most successful applications of neural nets. Investigations dealing with control of large space structures and with smart structures could also prove profitable.

REFERENCES

1. Rehak, D. R.; Thewalt, C. R.; and Doo, L. L.: Neural Network Approaches in Structural Mechanics Computations. Computer Utilization in Structural Engineering, ed. J. J. Nelson, Jr., ASCE PRoceedings from Structures Congress 1989.
2. McAulay, A. D.: Optical Neural Network for Engineering Design. NAECON, 1988.
3. Bennet, J.; Creary, L.; Engelmore, R.; and Melosh, R.: SACON: A Knowledge-Based Consultant for Structural Analysis. Stanford Heuristic Programming Project, Computer Science Department Report No. STAN-CS-78-699, September 1978.
4. Melosh, R. J.; Berke, L.; and Marcal, P. V.: Knowledge-Based Consultation for Selecting a Structural Analysis Strategy, NASA Conference Publication 2059, November 1978.
5. Rummelhart, D. E., and McCleland, J. L.: Parallel Distributed Processing, Volume 1: Foundations. The MIT Press, Cambridge, MA, 1988.
6. Yoh-Han Pao: Adaptive Pattern Recognition and Neural Networks. Addison-Wesley Publishing Co., 1989.
7. Baffes, P. T.: NETS 2.0 User's Guide. LSC-23366, NASA Lyndon B. Johnson Space Center, September 1989.

Table 1. Summary of Training Results with no Hidden Layer

Number of Training Sets	Network Description	Cycles of Training	Error Description
50	(5,2) - five areas as inputs, two vertical displacements as outputs.	1500	ϵ = 0.087 Error not decreasing
50	(10,2) - five areas and five reciprocal areas as inputs - two vertical displacements as outputs.	50000	ϵ = 0.03927 Error decreasing slowly
50	(15,2) - five areas and ten area products of type A_i A_j ($i \neq j$) as inputs - two vertical displacements as outputs.	50000	ϵ = 0.05235 Error decreasing slowly
50	(20,2) - five areas, five reciprocal areas and ten values of type $\sin(A_i/A_{max})$, $\cos(A_i/A_{max})$ used as input - two vertical displacements as output.	50000	ϵ = 0.0398 Error decreasing slowly

(xx,yy) denotes xx input nodes and yy output nodes.

Table 2. Optimal Design for Five Bar Truss Using Trained Neural Net for Analysis

Network Description		Design Variables					Objective Functions
		X_1	X_2	X_3	X_4	X_5	
(5-7-4) 200 training sets, all four output displacements mapped.	Initial	1.0	1.0	1.0	1.0	1.0	58.28
	Final	2.131	2.032	2.679	2.766	1.0	128.626
(5-7-4) 500 training sets, all four output displacements mapped.	Initial	1.0	1.0	1.0	1.0	1.0	58.28
	Final	1.952	2.013	2.763	2.760	1.0	127.759
(5-7-2) 100 training sets, two vertical displacements mapped.	Initial	1.0	1.0	1.0	1.0	1.0	58.28
	Final	1.535	1.778	2.29	2.265	1.0	107.56
(5-7-2) 100 training sets, two vertical displacements scaled as constraints and mapped.	Initial	1.0	1.0	1.0	1.0	1.0	58.28
	Final	1.505	1.584	2.138	2.211	1.0	102.399
	Exact Solution	1.5	1.5	2.121	2.121	1.0	100.0

(xx-yy-zz) denotes a three layer architecture with xx input layer nodes, yy hidden layer nodes, and zz output layer nodes.

Table 3. Optimal Design for Ten Bar Truss Using Trained Neural Net for Analysis

Design Variables	Network Description			
	(10-6-6-2)* 100 Training Sets Used in a Range of ±25% About Optimum	(10-6-6-2)* 400 Training Sets Used in a Range of ±25% About Optimum	(10-6-6-2) 100 Training Sets Used in a Range of 0.01-55.0 in^2 Output Scaled to Reduce Range of Variation	Solution From Exact Analysis
X_1	30.774	30.967	30.508	30.688
X_2	0.112	0.100	0.100	0.100
X_3	17.40	19.136	26.277	23.952
X_4	11.425	14.279	11.415	15.461
X_5	0.108	0.100	0.100	0.100
X_6	0.487	0.434	0.413	0.552
X_7	5.593	5.593	5.593	8.421
X_8	22.953	20.031	21.434	20.606
X_9	20.886	19.966	22.623	20.554
X_{10}	0.100	0.100	0.100	0.100
Objective Function	4692.49	4666.71	5010.22	5063.81

* Lower bound of design variables used as initial design - was infeasible.

Table 4. Ten Bar Truss Optimum Designs with H as Design Condition

H	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	WT
300	9.53	.1	9.67	4.73	.1	.1	6.34	6.15	6.15	.1	1749.6
305	9.37	.1	9.51	4.65	.1	.1	6.28	6.09	6.09	.1	1733.2
310	9.22	.1	9.36	4.57	.1	.1	6.22	6.03	6.03	.1	1717.4
315	9.07	.1	9.21	4.50	.1	.1	6.16	5.98	5.98	.1	1702.3
320	8.93	.1	9.07	4.43	.1	.1	6.11	5.92	5.92	.1	1687.9
325	8.79	.1	8.92	4.36	.1	.1	6.05	5.87	5.87	.1	1673.5
330	8.66	.1	8.79	4.29	.1	.1	6.00	5.83	5.83	.1	1660.9
335	8.53	.1	8.66	4.23	.1	.1	5.96	5.78	5.78	.1	1648.3
340	8.40	.1	8.54	4.16	.1	.1	5.92	5.73	5.73	.1	1635.9
345	8.28	.1	8.41	4.10	.1	.1	5.87	5.68	5.68	.1	1624.5
350	8.16	.1	8.29	4.05	.1	.1	5.82	5.65	5.65	.1	1613.5
355	8.05	.1	8.17	4.00	.1	.1	5.78	5.60	5.60	.1	1603.2
360	7.93	.1	8.06	3.93	.1	.1	5.74	5.68	5.68	.1	1593.1
365	7.83	.1	7.95	3.88	.1	.1	5.70	5.53	5.53	.1	1583.5
370	7.72	.1	7.84	3.83	.1	.1	5.66	5.50	5.50	.1	1574.3
375	7.61	.1	7.73	3.77	.1	.1	5.63	5.45	5.45	.1	1565.2
380	7.51	.1	7.63	3.73	.1	.1	5.60	5.42	5.42	.1	1557.1
385	7.42	.1	7.53	3.68	.1	.1	5.56	5.39	5.39	.1	1549.1
390	7.32	.1	7.44	3.63	.1	.1	5.53	5.36	5.36	.1	1541.4
395	7.23	.1	7.34	3.58	.1	.1	5.49	5.32	5.32	.1	1534.1
400	7.14	.1	7.25	3.54	.1	.1	5.46	5.30	5.30	.1	1527.1

Table 5. Training Accuracy with (1,6,3) Net

Training Pairs				Training Accuracy (RMS = 0.9%)				
Input	Output							
H	A1	A2	WT	A1	%	A2	WT	%
300	9.53	.1	1749.6	9.543	.14	.101	1744.4	.30
310	9.22	.1	1717.4	9.240	.23	.101	1721.0	.21
320	8.93	.1	1687.9	8.955	.28	.100	1693.4	.33
330	8.66	.1	1660.9	8.668	.11	.100	1665.6	.28
340	8.40	.1	1635.9	8.401	.01	.100	1640.3	.27
350	8.16	.1	1613.5	8.159	.012	.100	1618.1	.29
360	7.93	.1	1593.1	7.930	.00	.100	1597.2	.26
370	7.72	.1	1574.3	7.711	.12	.100	1577.2	.19
380	7.51	.1	1557.1	7.507	.07	.100	1558.2	.07
390	7.32	.1	1541.4	7.326	.08	.100	1542.1	.05
400	7.16	.1	1527.1	7.187	.38	.100	1529.3	.14

Table 6. Test Set of Neural Net Estimates

Input	Optimum			N-N Estimates				
H	A1	A2	WT	A1	%	A2	WT	%
305	9.37	.1	1733.2	9.364	.11	.101	1733.5	.00
315	9.07	.1	1702.3	9.097	.30	.100	1707.1	.28
325	8.79	.1	1673.5	8.801	.12	.100	1679.0	.33
335	8.53	.1	1648.3	8.529	.01	.100	1652.8	.27
345	8.28	.1	1624.5	8.283	.04	.100	1629.5	.31
355	8.05	.1	1603.2	8.043	.09	.100	1607.5	.27
365	7.83	.1	1583.5	7.818	.15	.100	1587.0	.22
375	7.61	.1	1565.2	7.604	.08	.100	1567.4	.20
385	7.42	.1	1549.1	7.361	.79	.100	1550.2	.07
395	7.23	.1	1534.1	7.252	.30	.100	1535.4	.08

Table 7. Ten Bar Truss Optimum Designs for Training with L_1, L_2, and H as Design Conditions

Input			Output									
L1	L2	H	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	WT
310	350	380	6.89	.1	7.00	3.62	.1	.1	5.24	5.08	5.35	.1	1356.7
345	326	360	7.40	.1	7.50	3.56	.1	.1	5.62	5.45	5.31	.1	1456.4
371	329	310	8.95	.1	9.10	4.17	.1	.1	6.33	6.14	5.74	.1	1684.5
360	300	340	7.69	.1	7.83	3.47	.1	.1	5.91	5.73	5.25	.1	1492.6
315	340	340	7.64	.1	7.76	3.93	.1	.1	5.53	5.36	5.56	.1	1407.6
380	355	390	7.47	.1	7.60	3.58	.1	.1	5.67	5.50	5.32	.1	1605.7
322	319	400	6.35	.1	6.46	3.13	.1	.1	5.21	5.05	5.03	.1	1314.2
400	300	400	9.25	.1	9.41	3.93	.1	.1	6.77	6.56	5.56	.1	1780.9
300	400	300	9.27	.1	9.39	5.25	.1	.1	5.73	5.57	6.57	.1	1593.8
311	350	315	8.33	.1	8.45	4.36	.1	.1	5.70	5.53	5.88	.1	1464.6

Table 8. Ten Bar Truss Optimum Designs for Checking Estimates of the Trained Neural Net

Input			Optimum Solutions										
L1	L2	H	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	WT
342	351	383	7.18	.1	7.29	3.60	.1	.1	5.44	5.27	5.34	.1	1466
360	360	360	7.93	.1	8.06	3.93	.1	.1	5.74	5.56	5.56	.1	1593
320	350	360	7.38	.1	7.50	3.82	.1	.1	5.43	5.26	5.49	.1	1417
340	370	340	8.29	.1	8.41	4.28	.1	.1	5.74	5.56	5.82	.1	1578
310	350	380	6.89	.1	6.99	3.62	.1	.1	5.24	5.08	5.35	.1	1356
345	326	360	7.39	.1	7.51	3.56	.1	.1	5.62	5.45	5.30	.1	1456
371	329	310	8.95	.1	9.16	4.17	.1	.1	6.33	6.14	5.74	.1	1684

Table 9. Trained Neural Net Estimates of Ten Bar Truss Optimum Designs

A1	7.138	8.000	7.365	8.398	6.908	7.310	8.916
%	0.580	0.780	0.200	1.300	0.260	1.080	0.380
A2	0.503	0.505	0.502	0.507	0.502	0.501	0.504
A3	7.245	8.160	7.485	8.558	7.013	7.448	9.089
%	0.620	1.240	0.200	1.760	0.330	0.820	0.770
A4	3.554	3.957	3.777	4.354	3.551	3.533	4.177
%	1.230	0.690	1.130	1.730	1.900	0.760	0.170
A5	0.499	0.499	0.500	0.499	0.500	0.500	0.500
A6	0.501	0.501	0.500	0.501	0.500	0.500	0.500
A7	5.456	5.783	5.461	5.781	5.309	5.586	6.359
%	0.290	0.750	0.570	0.710	1.320	0.610	0.460
A8	5.293	5.594	5.301	5.598	5.164	5.416	6.157
%	0.470	0.610	0.780	0.680	1.650	0.620	0.280
A9	5.326	5.574	5.477	5.863	5.336	5.311	5.740
%	0.260	0.250	0.240	0.740	0.260	0.210	0.000
A10	0.499	0.500	0.499	0.499	0.497	0.503	0.500
WT	1466.000	1598.000	1417.000	1585.000	1362.000	1451.000	1693.000
%	0.000	0.310	0.000	0.440	0.440	0.340	0.530

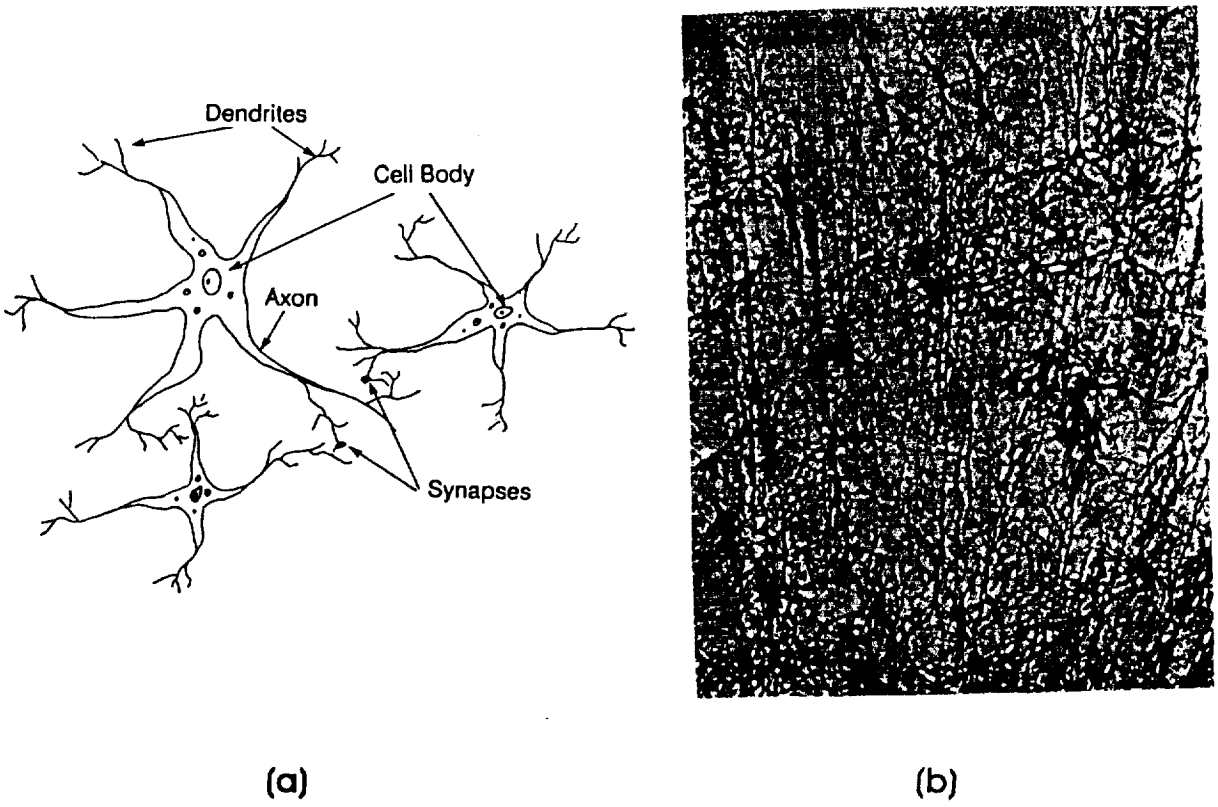

Figure 1. Biological Neuron and Neural Net

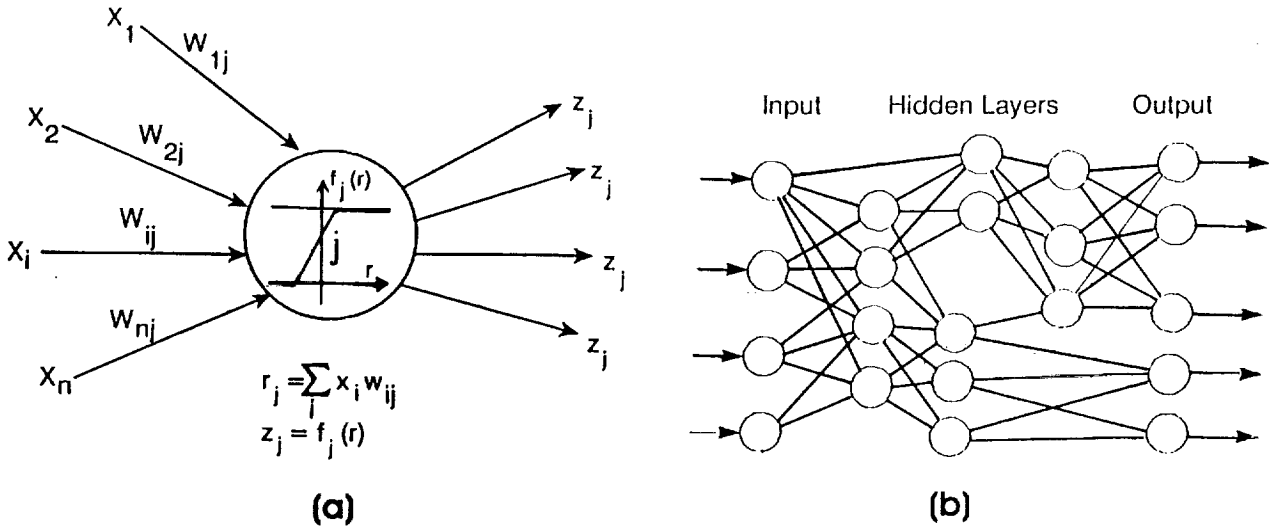

Figure 2. Artificial Neuron and Neural Net

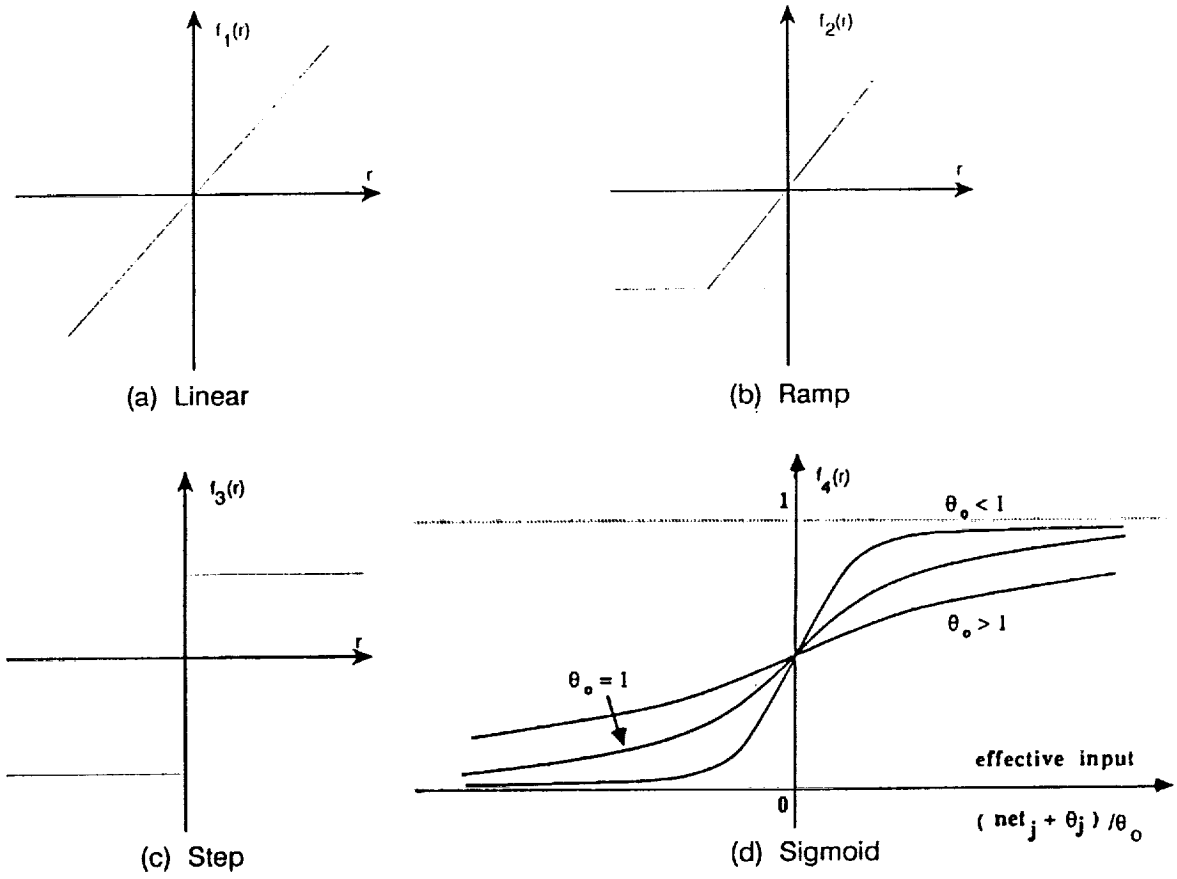

Figure 3. Activation Functions

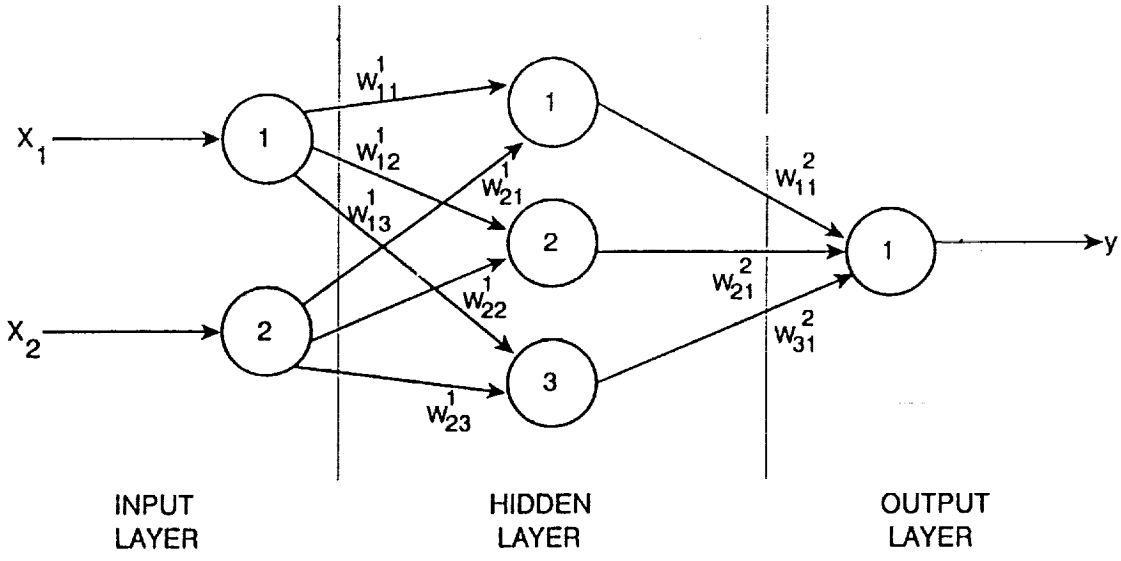

Figure 4. Six Neuron Neural Net with Hidden Layer

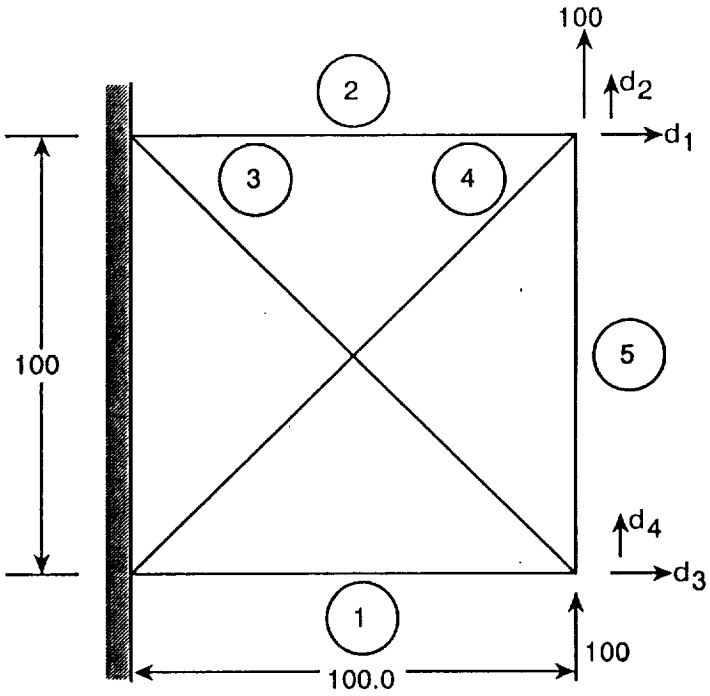

Figure 5. Five Bar Truss

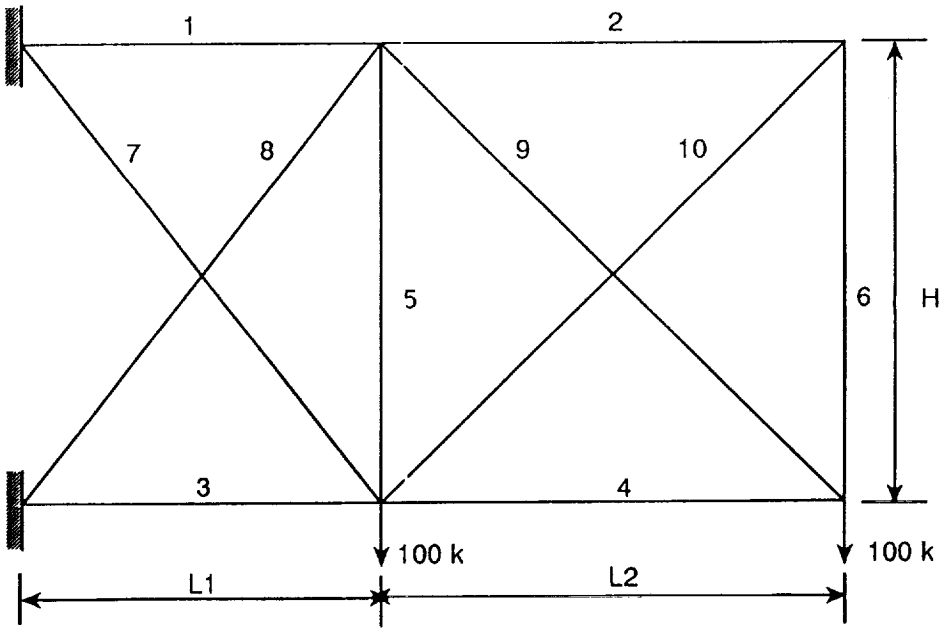

Figure 6. Ten Bar Truss

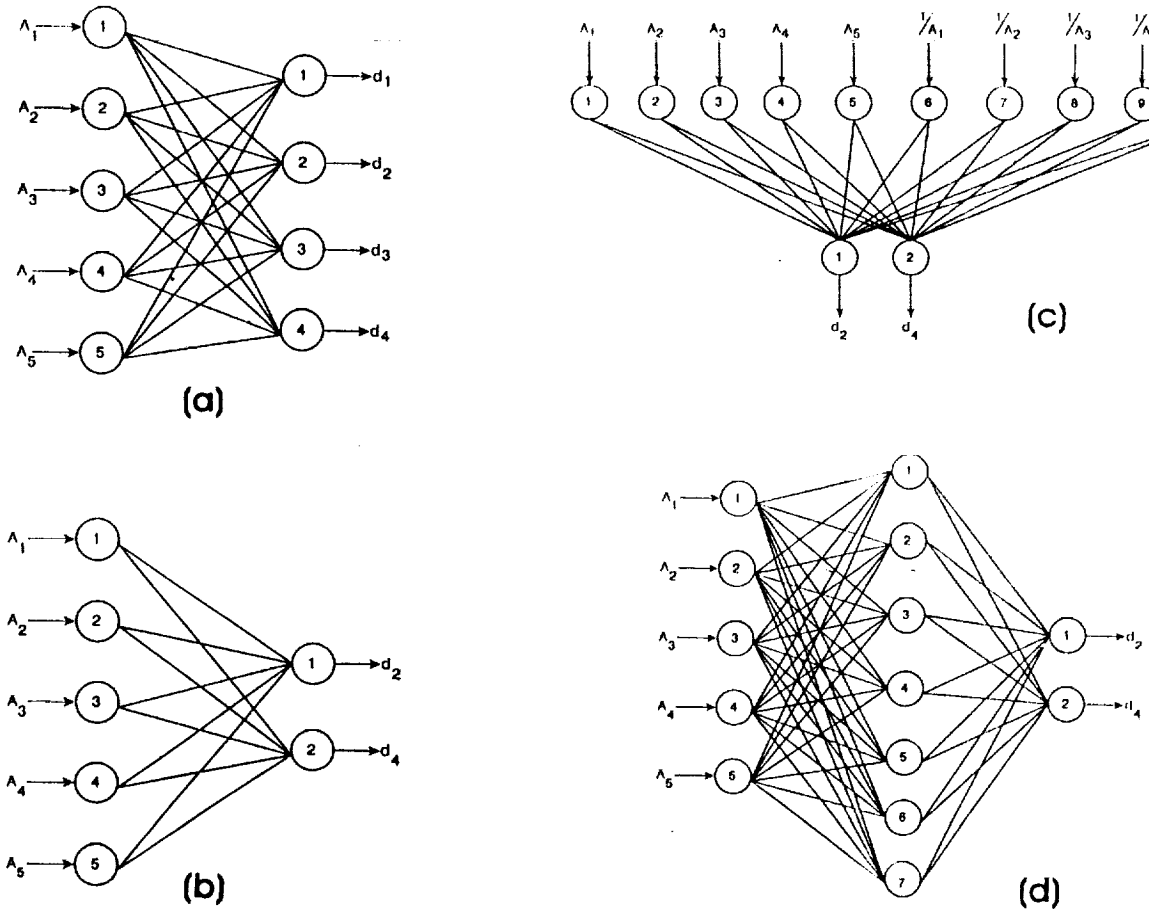

Figure 7. Neural Nets for Five Bar Truss Problem

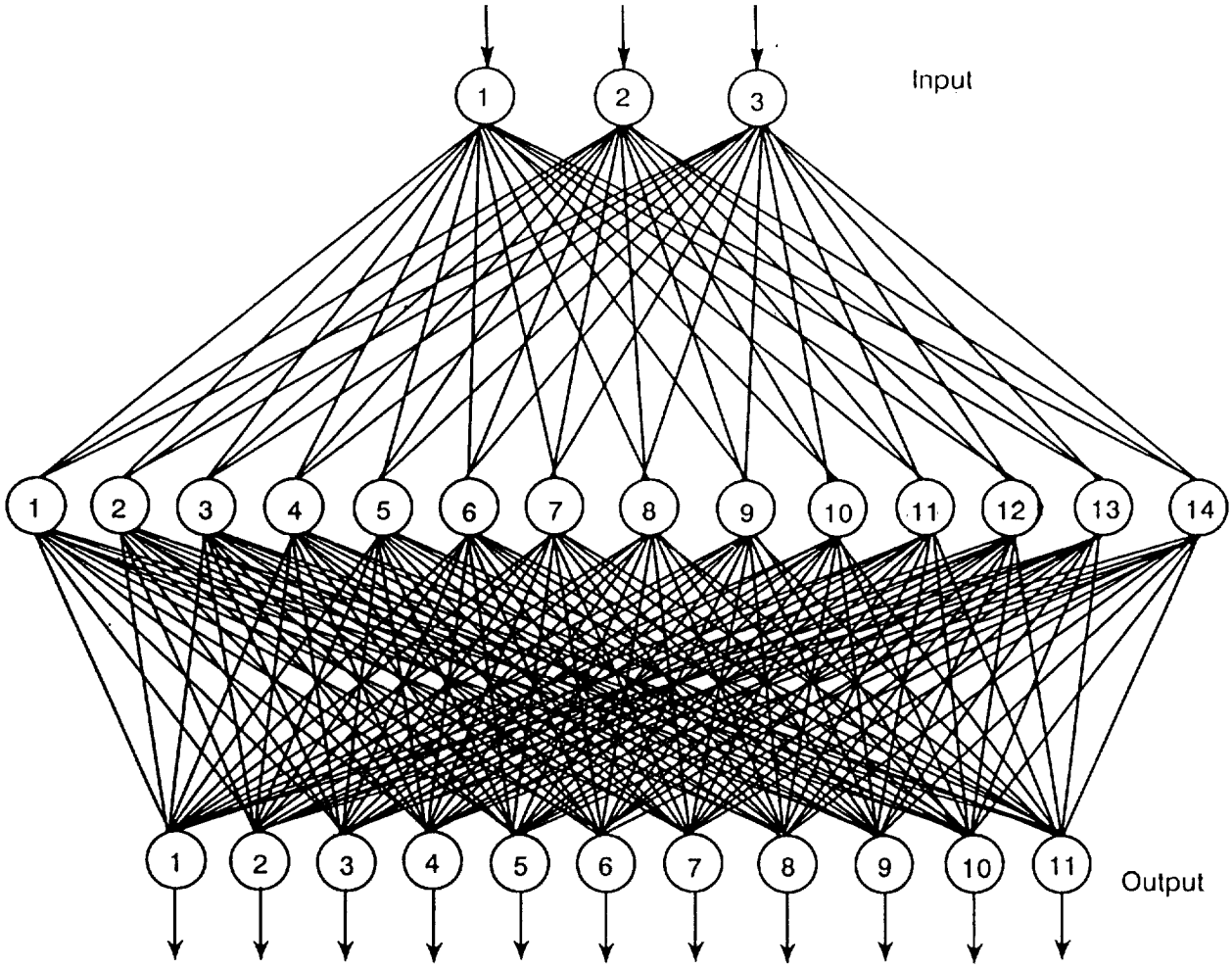

Figure 8. Neural Net for Ten Bar Truss Optimization Expert

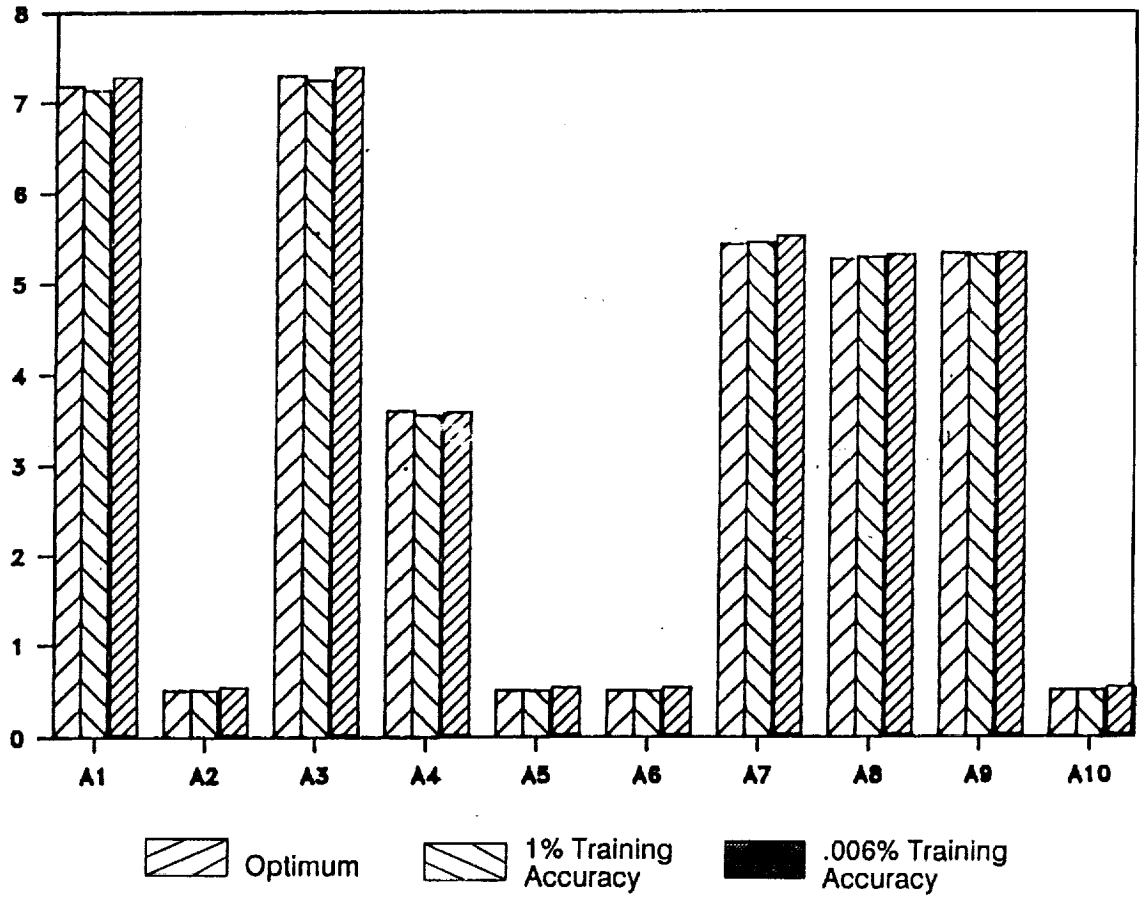

Figure 9. Optimum Design versus Neural Net Estimates.

NASA
National Aeronautics and Space Administration

Report Documentation Page

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
NASA TM-102420		

4. Title and Subtitle
Applications of Artificial Neural Nets in Structural Mechanics

5. Report Date

6. Performing Organization Code

7. Author(s)
Laszlo Berke and Prabhat Hajela

8. Performing Organization Report No.
E-5941

10. Work Unit No.
505-63-1B

9. Performing Organization Name and Address
National Aeronautics and Space Administration
Lewis Research Center
Cleveland, Ohio 44135-3191

11. Contract or Grant No.

13. Type of Report and Period Covered
Technical Memorandum

12. Sponsoring Agency Name and Address
National Aeronautics and Space Administration
Washington, D.C. 20546-0001

14. Sponsoring Agency Code

15. Supplementary Notes
Presented at the Lecture Series on "Shape and Layout Optimization of Structural Systems" at the International Centre for Mechanical Sciences, Udine, Italy, July 16-20, 1990. Laszlo Berke, NASA Lewis Research Center; Prabhat Hajela, Rensselaer Polytechnic Institute and Summer Faculty Fellow at NASA Lewis Research Center.

16. Abstract
A brief introduction to the fundamentals of Neural Nets is given first, followed by two applications in structural optimization. In the first case the feasibility of simulating with neural nets the many structural analyses performed during optimization iterations was studied. In the second case the concept of using neural nets to capture design expertise was investigated.

17. Key Words (Suggested by Author(s))
Structures
Optimization
Analysis
Neural Nets

18. Distribution Statement
Unclassified - Unlimited
Subject Category 39

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of pages	22. Price*
Unclassified	Unclassified	22	A03

NASA FORM 1626 OCT 86

*For sale by the National Technical Information Service, Springfield, Virginia 22161