# Backcalculation of Flexible Pavement Moduli Using Artificial Neural Networks

ROGER W. MEIER AND GLENN J. RIX

Artificial neural networks provide a fundamentally new approach to backcalculation of pavement layer moduli from falling-weight deflectometer deflection basins. An artificial neural network is a highly interconnected collection of simple processing elements that can be trained to approximate a complex, nonlinear function through repeated exposure to examples of the function. In the context of backcalculation, a neural network can be trained to approximate the inverse function by repeatedly showing it forward problem solutions. The single most important advantage of using neural networks for backcalculation is speed. Neural networks trained in this study are more than three orders of magnitude faster than conventional gradient search algorithms. Such speed makes real-time backcalculation of moduli possible. Two backpropagation neural networks were trained to backcalculate pavement moduli for three-layer flexible pavement profiles. Synthetic deflection basins with a wide variety of layer moduli and thicknesses were used to train both networks. One network was trained using ideal deflection basins. Subsequent testing showed that the network could backcalculate pavement layer moduli accurately. A second network was trained using basins, with random noise added to simulate measurement errors. When tested using similarly noisy deflection basins, that network did a reasonably good job of predicting moduli, although it exhibited much more scatter in the results. That same network performed very well on experimental data from two pavement test sections of the Strategic Highway Research Program.

The falling-weight deflectometer (FWD) is used widely to nondestructively assess the structural properties of flexible pavements. Evaluation of FWD test results entails backcalculating in situ pavement layer moduli from measured deflections. It is usually accomplished by matching theoretical and experimental deflection basins. Theoretical deflection basins commonly are calculated using static, multilayer, linear-elastic analyses. In principle, it is also possible to use algorithms that account for dynamic effects and nonlinear material behavior, but they involve significantly greater computation times, which makes them unacceptable for production use.

Current basin-matching programs fall into two broad groups. Most programs employ gradient search techniques to adjust the pavement layer moduli iteratively until the theoretical and experimental deflection basins agree within a specified tolerance. The __DEF series of programs (*1*) is typical of the approach. Required inputs include experimental deflection measurements and pavement layer thicknesses. The iterative solution technique also requires an initial estimate of the solution (seed moduli) and a range of moduli to constrain the solution.

A second approach is to interpolate within a data base of theoretical basins. The MODULUS program (*2*) is an example of this approach. A data base of theoretical basins is generated for prescribed pavement layer thicknesses by parametrically varying the pavement layer moduli within expected ranges. Once the data base is complete, MODULUS uses the Hooke-Jeeves pattern-searching algorithm to choose the deflection basins in the data base that most closely match the experimental basin. MODULUS then calculates the layer moduli corresponding to the experimental basin using Lagrange interpolation. Besides deflection measurements, MODULUS requires a range of moduli for the surface layer and base course and an initial estimate of the subgrade modulus.

The authors of this paper present a fundamentally different approach to FWD backcalculation by using artificial neural networks. Artificial neural networks have been used to solve problems involving pattern recognition, classification, and mapping (*3*). The class of neural networks known as backpropagation networks is universal functional approximators (*4*) that can "learn" a functional mapping through repeated exposure to examples of that mapping. In the context of FWD analysis, a backpropagation neural network can be "trained" to map deflection basins onto their corresponding pavement layer moduli. The best way to train such a network is to use experimentally determined deflection basins along with independently measured pavement layer moduli. Lacking sufficient quantities of such data over a broad range of layer moduli and thicknesses, synthetic deflection basins can be obtained by solving the forward problem with many different combinations of pavement layer properties. A neural network can then be taught to map these synthetic deflection basins back onto their corresponding layer moduli. The latter approach is taken in this paper.

There are several advantages to using neural networks for FWD analysis. The mathematical simplicity of neural networks makes them computationally efficient. They make real-time backcalculation of moduli possible using personal computers. Unlike other backcalculation techniques, a neural network does not require seed moduli or moduli ranges. That eliminates the subjectivity associated with choosing seed moduli and allows the backcalculation procedure to be automated for use by less experienced operators. Furthermore, because a neural network does not explicitly match deflection basins, the pavement moduli determined by the neural network are independent of the error measures (e.g., mean-squared error, maximum absolute error) and the tolerance criteria used to determine convergence.

## ARTIFICIAL NEURAL NETWORKS

Artificial neural networks are biologically inspired analogues of the human brain. They are composed of a great many operation-

R. Meier, U.S. Army Engineer Waterways Experiment Station, Geotechnical Laboratory, Mobility Systems Division, 3909 Halls Ferry Road, Vicksburg, Miss. 39180-6199. G. Rix, Georgia Institute of Technology, School of Civil Engineering, Atlanta, Ga. 30332-0355.

ally simple but highly interconnected processing units. The processing units themselves have certain functional similarities to biological neurons, and their organization bears at least superficial resemblance to the organization of neurons in the brain.

Artificial neural networks exhibit many characteristics of the human brain (*5*). For example, certain types of neural networks will "teach themselves," through repeated exposure to a set of data, to recognize common features within the data and to group the data accordingly. Other types of neural networks can be programmed to associate a set of input patterns with their respective output patterns. Artificial neural networks can also generalize an ideal mapping from imperfect examples and extract essential information from input containing both relevant and irrelevant data. Their ability to "see" through noise and distortion to the underlying pattern has been exploited successfully for solving many problems related to pattern recognition.

The most common network architecture used for mapping, classification, and forecasting problems is the multilayer, feed-forward network (*6*). Such networks consist of several layers of processing elements (Figure 1). The processing elements pass information in the form of signal patterns from the input layer of the network through a series of hidden layers to the output layer. Signals travel between processing elements along connections whose strengths can be adjusted to amplify or attenuate the signal as it propagates. Each processing element sums the impinging signals to determine a net level of excitation. A nonlinear activation function provides a graded response to that excitation. The element then passes on the response to each of the processing elements in the next layer (Figure 2). The distribution of connection strengths throughout the network uniquely determines the output signal pattern that results from a given input signal pattern. In that respect, the connection strengths store the "knowledge" contained in the network.

The excitation level of a processing element is modeled mathematically as a weighted sum of its inputs:

$$N_j = \sum_{i=1}^{n} w_{ji} x_i \tag{1}$$

where $x_i$ is the signal coming from the $i$th processing element in the preceding layer, and $w_{ij}$ is the weight assigned to that connection. The weights determine the degree of signal amplification or attenuation on the incoming connections.

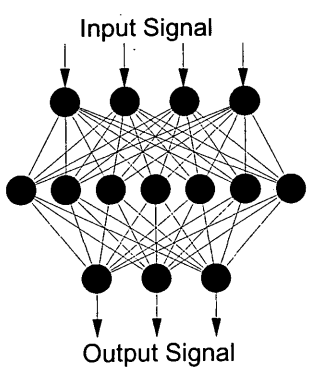


**FIGURE 1 Artificial neural network architecture.**

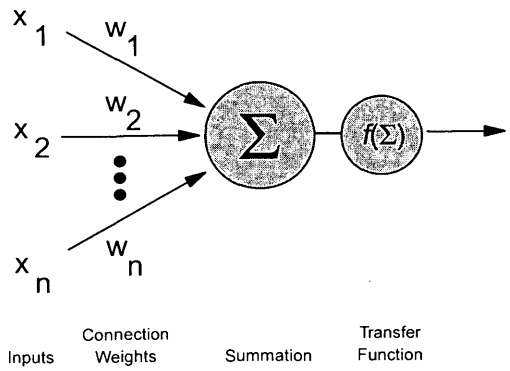


**FIGURE 2 Basic processing element for multi-layer, feed-forward network.**

The processing element's response to the net excitation $N_j$ commonly is modeled by the sigmoidal logistic function:

$$a_j = f(N_j) = \frac{1}{1 + e^{-N_j}} \tag{2}$$

The function accepts input over the range $(-\infty, \infty)$ and uniquely maps it into the range [0,1]. That not only prevents the signals from growing unbounded as they are summed repeatedly and passed on, but it also introduces nonlinearity into the network. Without this nonlinearity, the network would simply output linear combinations of the input signals. That would severely limit the type of mapping it could perform.

A neural network gains its knowledge through training. A supervised learning method commonly is used to train feed-forward networks. In supervised learning, a set of training data (consisting of pairs of input-output patterns exemplifying the mapping to be learned) is presented to the network, one example at a time. For each example, the input pattern is propagated through the network, and the resulting output pattern is compared with the target output. A learning algorithm is employed to adjust incrementally the connection weights in order to reduce the difference between calculated and target output. The ability to self-adjust is an essential feature of neural computing. It would be impossible to establish manually the connection weights needed to perform any but the simplest of mappings.

Multilayer, feed-forward networks commonly are trained by a technique known as error backpropagation. After each training example is presented to the network, the differences between the calculated and target output patterns are computed and propagated backward through the network according to the existing network connection weights. Individual connection weights then are adjusted in the direction that reduces the error apportioned to them. If training is successful, connection weights attain values that globally minimize the output error (commonly expressed as either the root-mean-square or arithmetic mean) for all the inputs in the training set.

The most common learning algorithm used in backpropagation networks is the generalized delta rule (*7,8*). The generalized delta rule is essentially a gradient descent scheme that seeks a global minimum of the error surface that relates the output errors to the connection weights. In the simplest form of the generalized delta

rule, weight changes at each step in the gradient descent are calculated as follows:

$$\Delta w_{ij} = \alpha \nabla E(w_{ij}) \tag{3}$$

where $\nabla E(w_{ij})$ is the gradient of the error surface with respect to the weight in question, and $\alpha$ is the "learning rate." The learning rate regulates the step size of the gradient descent. A more advanced form uses an additional momentum term to help the gradient descent avoid shallow local minima:

$$\Delta w_{ij}(t) = \alpha \nabla E(w_{ij}) + \beta \Delta w_{ij}(t - 1) \tag{4}$$

where $\Delta w_{ij}(t - 1)$ and $\Delta w_{ij}(t)$ are the weight changes applied on successive steps, and $\beta$ regulates the amount of momentum.

Invoking the chain rule of differentiation, the gradient of the error surface with respect to an individual connection weight, $w_{ij}$, instead can be expressed as

$$\nabla E(w_{ij}) = \frac{\partial E(w_{ij})}{\partial N_j} \frac{\partial N_j}{\partial w_{ij}} = \delta_j \frac{\partial N_j}{\partial w_{ij}} = \delta_j a_i \tag{5}$$

where the $\delta_j$ (from which the generalized delta rule takes its name) are the gradients of the error surface with respect to the net excitation level of each processing element, and the $a_i$ are the individual inputs to each processing element. At the output units, the $\delta_j$ are computed as the product of the output error and the derivative of the activation function:

$$\delta_j = (t_j - a_j) \frac{df(N_j)}{dN_j} \tag{6}$$

where $t_j$ is the target output. One of the reasons the sigmoidal logistic function is so popular as an activation function is that the derivative can be calculated easily:

$$\frac{df(N_j)}{dN_j} = a_j(1 - a_j) \tag{7}$$

At the processing elements in the other network layers, the target outputs are not known a priori. Instead, the errors attributed to those processing elements are estimated by assessing each element's relative contribution to the outputs and, thus, the errors of the elements in the succeeding layer:

$$\delta_j = \frac{df(N_j)}{dN_j} \sum_{k=1}^{n} \delta_k w_{jk} \tag{8}$$

By working backward from the output layer, errors can be apportioned successively to the processing elements in the remaining layers of the network.

Once trained, the network will provide an approximate functional mapping of any input pattern onto its corresponding output pattern. This process is extremely fast because the input pattern is propagated once through the network, a task that involves passing only weighted sums through the sigmoidal logistic function.

The authors implemented the algorithm described above in a FORTRAN computer program. The program logic is summarized in Figure 3. The implementation is relatively straightforward because a neural network gains its processing power from its highly interconnected architecture, not from mathematical complexity.

## FWD BACKCALCULATION USING ARTIFICIAL NEURAL NETWORKS

### Initial Network Training

Backpropagation neural networks are universal approximators; their training times increase rapidly with increasing problem complexity, which places some practical limit on the mappings that they can learn. Instead of trying to train a network to handle a variable number of pavement layers, the authors chose to train a neural network to backcalculate moduli for a three-layer profile consisting of an asphaltic concrete (AC) surface layer, an unstabilized base course, and a subgrade. Assumed ranges of the layer properties are indicated in Table 1. Thickness of the subgrade was arbitrarily assigned a value of 30.4 m (100 ft) to eliminate the influence of the rigid layer, resulting in an essentially infinite subgrade (*9*). The authors attempted to cover a broad range of realistic layer properties. If the anticipated layer properties were substan-

```
Read in the training parameters and network dimensions
Initialize the connection weights to small random numbers
For each training epoch:
    For each input/output pair in the training set:
        Propagate the input through the network (Eqs. 1,2)
        Compute the deltas for the output layer (Eqs. 6,7)
        Compute the deltas for the remaining layers (Eqs. 7,8)
        Compute the weight changes for all of the layers (Eqs. 4,5)
    For each input/output pair in the testing set:
        Propagate the input through the network (Eqs. 1,2)
        Compute output errors and update output error statistics
Report on the training progress
Close all input and output files
```

**FIGURE 3 Computer implementation of backpropagation algorithm.**

**TABLE 1 Pavement Layer Properties Used To Train Neural Networks**

| Layer | Layer Modulus (MPa)[a] | Layer Thickness (cm)[b] | Poisson's Ratio |
|---|---|---|---|
| Asphalt | 1725 - 20,685 | 5 - 30 | 0.325 |
| Base | 35 - 1035 | 15 - 75 | 0.35 |
| Subgrade | 35 - 345 | 3050 | 0.35 |

[a] 1 MPa = 0.145 ksi
[b] 1 cm = 0.394 in

tially different from those used here, another neural network would have to be trained.

A training set of 10,000 synthetic deflection basins was generated using the static, multilayer, linear-elastic program WESLEA (*10*). For each deflection basin, the thicknesses and moduli of the AC and base layers and the modulus of the subgrade were selected randomly from uniform distributions within the limits identified in Table 1. Pavement deflections were calculated for a dynamic load of 40 kN (9,000 lb) acting over an area with a radius of 15 cm (5.91 in.). The authors assumed a fixed sensor spacing to reduce further the complexity of the mapping to be learned. Initial experimentation revealed that the Strategic Highway Research Program (SHRP) sensor spacings of 0, 20, 30, 45, 60, 90, and 150 cm (0, 8, 12, 18, 24, 36, and 60 in.) provided the network with more information about the AC and base moduli than did a uniform 30-cm (12-in.) spacing and allowed the network to make better predictions. SHRP spacing therefore was used exclusively.

If conventional backpropagation networks are used, the network architecture must be established before the start of training. At a minimum, the network must have an input layer and an output layer. The number of neurons in those layers is easy to determine: they are equal to the number of input and output parameters, respectively. There are, however, no well-established procedures for choosing the number of hidden layers nor the number of neurons in each hidden layer (*11*). As a result, trial and error must be used to determine the optimum network architecture, which must strike a balance between insufficient knowledge capacity (too few connections) and excessive capacity. If the network has insufficient capacity, it will be incapable of accurately performing the required mapping. On the other hand, if the network has excessive capacity, it will in essence "memorize" the training examples. In that case, the network will be incapable of performing mappings for deflection basins that it has not memorized.

In principle, it is possible to approximate any functional mapping with a network consisting of one hidden layer (*4*). In practice, however, two hidden layers often allow the same functional mapping to be learned with fewer neurons. After experimenting with several different architectures, the authors chose the network architecture represented in Figure 4. The first hidden layer (closest to the input layer) contained 11 processing elements and the second contained eight processing elements. The input layer of that network contained nine processing elements (corresponding to the AC and base layer thicknesses and the seven deflections), and the output layer contained three processing elements (corresponding to the AC, base, and subgrade moduli).

Training proceeded by iteratively prsenting training examples to the network. Each pass through the set of 10,000 examples constituted a training epoch. During each epoch, the first 9,750

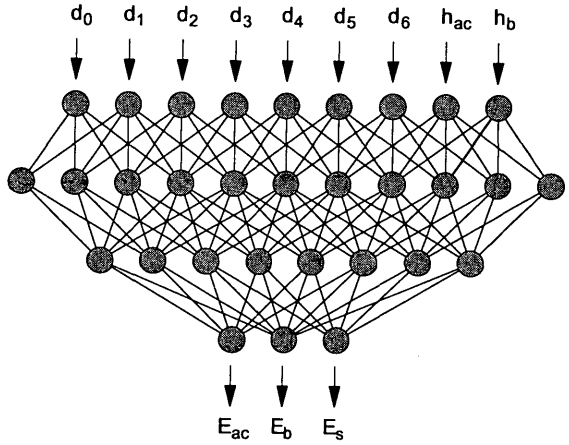


**FIGURE 4 Neural network architecture used for backcalculating pavement layer moduli from synthetic deflection basins.**

examples were used to train the network. The remaining 250 examples were used to test the network to monitor its training progress. (Neural networks should never be tested with the same data that are used to train them. It is important that the network be able to generalize beyond the training examples, instead of simply memorizing them.) At first, the mean squared error of the outputs drops rapidly as the training epochs are completed, as indicated in Figure 5(a). With further training, the output error asymptotically approaches some minimum level. Training of the network continued until it was clear that this level substantially had been reached.

Figure 5(b), (c), and (d) are scatter plots that compare the target and computed moduli of the asphalt, base, and subgrade layers, respectively, for the 250 test basins. The plots clearly show that the network successfully learned to backcalculate pavement layer moduli from synthetic deflection basins for the entire range of pavement layer properties included in the training set. In a broader context, these results are significant because they indicate that neural networks can be taught to solve complex, nonlinear inverse problems using training data generated by solving the forward problem.

## Increasing Network Robustness

Accurate deflection basin measurements are essential if backcalculated layer moduli are to be correct. However, it is unrealistic to expect field measurements to be perfectly accurate. Two pri-

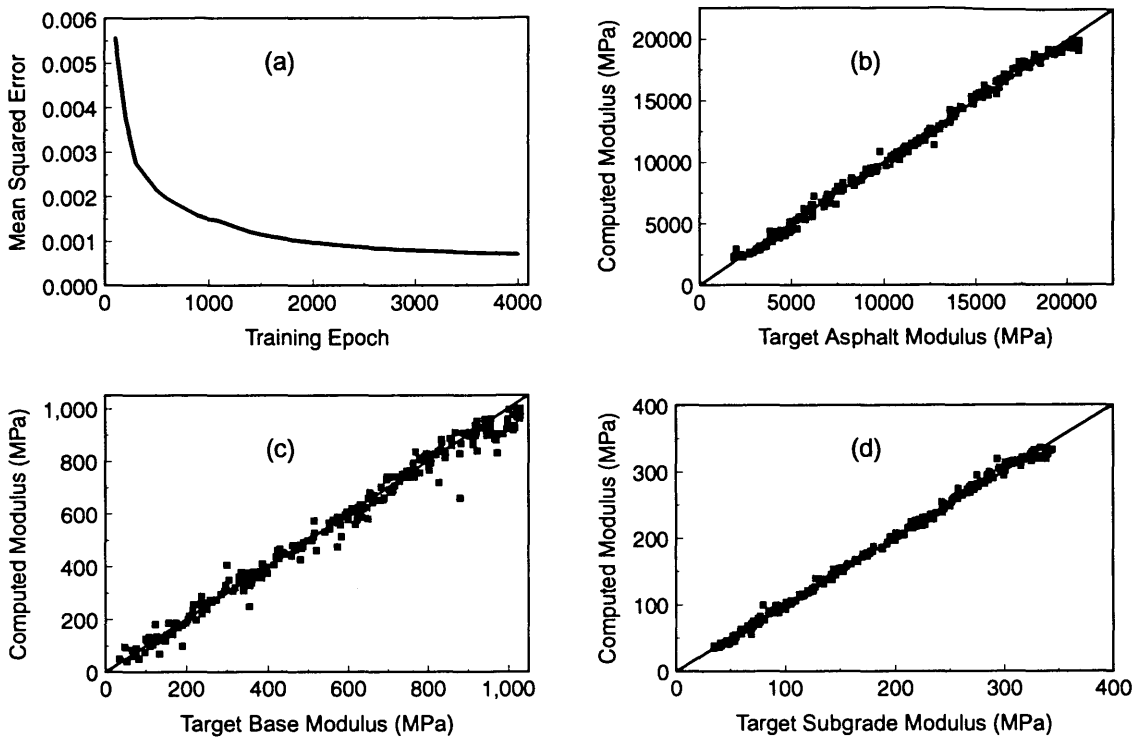


**FIGURE 5** *(a)* **Training progress; moduli for network trained with synthetic data, including test results for** *(b)* **asphalt,** *(c)* **base, and** *(d)* **subgrade.**

mary sources of deflection measurement error exist: systematic errors and repeatability errors. Typical specifications for the FWD test (*12*) require a systematic error no greater than 2 percent of the measured deflection and a repeatability error no greater than 2 μm (0.08 mil). Periodic calibration of the velocity transducers can minimize the systematic error, but repeatability errors are random.

One approach to dealing with errors is to include random noise in the deflection basins that are used to train a network—a technique known as noise injection (*13*). Including random noise in the training data makes the network more robust because it learns to produce reasonably accurate moduli in the presence of noise. The authors trained a robust version of the network by adding random noise to each of the seven deflections in each training example just before presenting it to the network. In this way, even though the training basins were reused for each epoch, the added noise was different every time. The random variates were drawn from uniform distributions whose limits were equal to the larger of ± 2 percent of the ideal deflection or ± 2.5 μm (± 0.1 mil). The latter was made slightly larger than the test specification to permit some room for error. Because the task of learning to map noisy data is more difficult for the network, the authors arbitrarily increased the number of processing elements in both hidden layers to 15 before the start of training.

The training progress of the robust network is presented in Figure 6(a). Note that the final value of mean squared error is about 0.0055 for the robust network compared with 0.0007 for the network trained with noise-free data, as seen in Figure 5(a). A tradeoff between accuracy and robustness is to be expected. Also note that the network required about twice as many epochs of training (8,000 versus 4,000) to achieve a nearly constant mean squared error. That is also to be expected because the technique used to generate the random noise ensured that the network never saw the same basin twice, whereas the network trained using ideal deflection basins saw each of those basins 4,000 times.

To assess the robust network's backcalculation abilities, the 250 deflection basins used to test the original network were also modified by adding random noise to the deflection measurements. Tests to determine the repeatability of FWD measurements (*14*) have shown that individual transducers have a standard deviation of ± 1.95 μm. Because the error is random, it can be lessened by replicating the test and averaging the results. Irwin et al. (*14*) recommend that three to five replicates be conducted for each test. Therefore, the amount of noise added to each deflection is established by averaging five random variates drawn from a Gaussian distribution with a mean of zero and a standard deviation that is rounded off to ± 2 μm (0.08 mil). Because the random variates were drawn from a Gaussian distribution instead of the uniform distribution used to train the network, it is possible that some of the test basins contained more noise than was present in the training set.

Figures 6(b), 6(c), and 6(d) compare the target and computed moduli of the asphalt, base, and subgrade layers, respectively, for the 250 modified test basins. The fine dotted lines included in those figures indicate the 95 percent prediction interval associated with a linear regression of the calculated moduli on the target moduli. Those are the bounds within which 95 percent of all future predictions should lie. There is a very good linear correlation between the predicted and target moduli: the $R^2$ values for the linear regressions are 0.961, 0.918, and 0.995 for the asphalt, base,

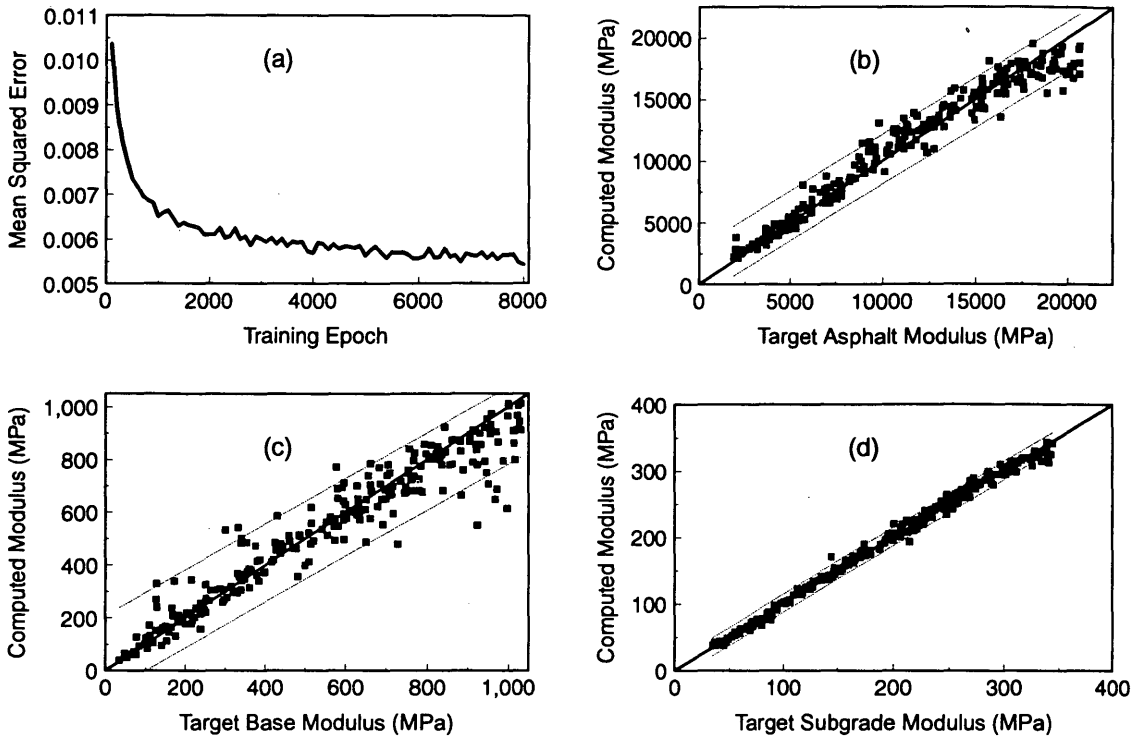


**FIGURE 6** (*a*) **Training progress; moduli for network trained with noise injection, including test results for** (*b*) **asphalt,** (*c*) **base, and** (*d*) **subgrade.**

and subgrade, respectively. The precision of results is, however, much lower than it is for the ideal deflection basins. There are more elaborate training algorithms and more complex network architectures that researchers could employ in the future to achieve better precision in the presence of noise.

## Backcalculation of Experimental Data

The neural network's performance on experimental deflection basins was evaluated using data from two SHRP pavement test sections described by Rada et al. (*15*). Pavement Sections A and B were selected because they were considered similar to the three-layer, flexible pavements used to train the neural network. Pavement profiles for Sections A and B are presented in Table 2. In Section A, the crushed limestone base and the soil aggregate subbase were combined to form a single base layer. In both sections, the subgrade was assumed to be semiinfinite because Rada et al. (*15*) report that no bedrock was encountered in either section within the top 6.5 m (20 ft).

Pavement deflections were normalized to a load of 40 kN (9,000 lb) and propagated through the robust network. The same deflections and layer thicknesses were used in MODULUS 4.0 (*2*) and WESDEF (*10*) and the results compared (Table 2). The asphalt and base moduli calculated by the neural network were similar to those computed by MODULUS 4.0. The WESDEF program predicted higher asphalt moduli and lower base moduli. All three programs predicted virtually identical subgrade moduli. Because the true moduli at the two test sections are not known, success can only be measured in comparison to the predictions produced by other programs. The similarity of the neural network moduli to those predicted by MODULUS 4.0 (which uses the data base approach) and WESDEF (which uses the gradient-search approach) is taken as an indication that the neural network performed well on these experimental data.

## Comparison of Processing Times

An advantage of using artificial neural networks is the speed at which pavement moduli can be backcalculated. Table 3 shows the processing times required for the trained neural network to backcalculate moduli for the 250 synthetic testing basins, with and without the addition of random noise. Also included in Table 3 are the times required by WESDEF to analyze the same 250 basins. Convergence criteria in WESDEF were adjusted to yield predictions as accurate as those of the neural network (Figures 5 and 6). Timing comparisons were conducted on a 33-MHz 80486 personal computer.

For basins with no random noise added, the neural network backcalculated moduli for all 250 basins in 0.9 sec. WESDEF required 25 min to complete the task. With random noise ($\pm 1.95$ μm) added to the synthetic deflections, WESDEF required 37.5 min. The artificial neural network processed "noisy" data as quickly as noise-free data because deflection inputs were simply propagated through the network. WESDEF, on the other hand, had to seek iteratively a theoretical basin to match the noisy experimental basin. The authors know of no other backcalculation algorithm that has the neural network's ability to backcalculate pavement moduli in real time.

TABLE 2 Comparison of Backcalculated Moduli for SHRP Pavement Test Sections

| Section | Layer | Layer Thickness (cm)[b] | Backcalculated Moduli (MPa)[a] Artificial Neural Network | MODULUS 4.0 | WESDEF |
|---|---|---|---|---|---|
| A | Asphalt | 12.6 | 8922 | 8619 | 11570 |
| | Base[c] | 64.5 | 290 | 283 | 221 |
| | Subgrade | Semi-Infinite | 221 | 207 | 228 |
| B | Asphalt | 10.7 | 5895 | 6350 | 10343 |
| | Base | 12.7 | 365 | 386 | 138 |
| | Subgrade | Semi-Infinite | 186 | 186 | 200 |

[a]1 MPa = 0.145 ksi
[b]1 cm = 0.394 in
[c]Combination of crushed limestone base and soil/aggregate subbase

Another advantage of a neural network is that the creation of the training data and the training of the network are completely separate from the use of the trained network. Thus, it is possible to train a network to account for dynamic effects and nonlinear material behavior without increasing its processing time. Although it will take significantly longer to create the training set, and slightly longer to train the network because of the increased complexity of the mapping, the trained network will backcalculate moduli as quickly as one trained using a static, multilayer, linear-elastic solution. That is in marked contrast to gradient-search programs that must repeatedly solve the more-complex dynamic and nonlinear forward problem to obtain an answer.

## FUTURE CONSIDERATIONS

This research is a first step in the development of a real-time backcalculation procedure for the FWD test that accounts for nonlinear material behavior and the dynamic nature of the test. The applicability of the present neural network is limited by the range of pavement layer properties included in the training set. Networks capable of operating across a broader spectrum of field conditions than were addressed here are certainly feasible but will require a more diverse training set. For this feasibility study, a brute-force approach using a large number of randomly generated profiles was adopted. The authors anticipate that a comprehensive training set can be developed without increasing the number of training examples, by using more refined methods of parameter variation. They also anticipate that network training can be accelerated despite a broader scope by using second- and third-generation training algorithms.

## SUMMARY AND CONCLUSIONS

Artificial neural networks provide a fundamentally different way to backcalculate pavement layer moduli from FWD deflection basins. Unlike conventional approaches that backcalculate moduli by trying to match theoretical and experimental deflection basins, a neural network simply maps deflection basins into their corresponding layer moduli. The network learns this functional mapping by adjusting the connection weights between its processing elements during repeated exposure to a set of examples (training data). In this study, the training data consisted of synthetic deflection basins generated for a wide range of pavement layer thicknesses and moduli using WESLEA.

Two backpropagation neural networks were successfully trained to backcalculate moduli for three-layer, flexible pavement systems. The first network was trained using synthetic basins with no random noise added. After training, the network was capable of backcalculating layer moduli with excellent accuracy. This initial result is important because it illustrates that a neural network can learn to solve an inverse problem by training it using forward problem solutions. A second network was trained using deflection basins with random noise added to simulate measurement errors. By using random noise in the training data, a final network should be most robust (i.e., it should provide reasonable estimates even for imperfect data). Although the calculated moduli contained

TABLE 3 Comparison of Processing Times To Backcalculate 250 Deflection Basins[a]

| | Noise-Free Deflection Basins | Basins with ±1.95μm of Random Noise |
|---|---|---|
| Neural Network | 0.9 sec | 0.9 sec |
| WESDEF[b] | 25 min | 37.5 min |

[a]Processing times measured on a 33-MHz 80486 personal computer
[b]Using sum of absolute percentage differences less than 3.5% as the convergence criterion and a 20-iteration limit

more scatter than the noise-free results, the estimates from the network trained and tested with noisy data were still reasonably accurate. The neural network trained with noise injections also backcalculated moduli that were similar to those predicted by MODULUS 4.0 for two SHRP pavement test sections.

Artificial neural networks offer several advantages for the backcalculation of moduli. The most important one is speed. Neural networks trained in this study are 1,500 to 2,200 times faster than a conventional gradient search technique. Such speed makes it possible to determine moduli in real-time on personal computers. Neural networks also eliminate the need for the user to specify seed moduli and moduli ranges. Without seed moduli and moduli ranges, backcalculations are less dependent on subjectivity introduced by the user.

The most promising aspect of neural networks is the ability to use more complex and realistic pavement and material models as the basis for a backcalculation. Solving the forward problem to create a training set is completely separate from use of the trained network for backcalculation. That means a neural network can be trained to account for dynamic and nonlinear material behavior and still be able to backcalculate moduli in real time.

## ACKNOWLEDGMENTS

Unless otherwise noted, information presented here was obtained from research conducted by the U.S. Army Engineer Waterways Experiment Station. The views of the authors do not purport to reflect the position of the Department of the Army or the Department of Defense. The second author would like to acknowledge the financial support provided by the Waterways Experiment Station. Both authors are grateful to Albert J. Bush III and to Don R. Alexander of the Waterways Experiment Station for their advice and support.

## REFERENCES

1. Bush, A. J., III. *Nondestructive Testing for Light Aircraft Pavements, Phase II: Development of the Nondestructive Testing Methodology.* Report FAA-RD-80-9-II, FAA, U.S. Department of Transportation, 1980.
2. Uzan, J., R. L. Lytton, and F. P. Germann. General Procedure for Backcalculating Layer Moduli. In *Nondestructive Testing of Pavements and Backcalculation of Moduli* (A. J. Bush III and G. Y. Baladi, eds.), ASTM 1026, ASTM, Philadelphia, Pa., 1989, pp. 217–228.
3. Dayhoff, J. *Neural Network Architectures: An Introduction.* Van Nostrand Reinhold, New York, 1990.
4. Hornik, K., M. Stinchcomebe, and H. White. Multilayer Feedforward Networks Are Universal Approximators. *Neural Networks,* Vol. 2, 1989, pp. 359–366.
5. Wasserman, P. D. *Neural Computing: Theory and Practice.* Van Nostrand Reinhold, New York, 1989.
6. Hecht-Nielsen, R. *Neurocomputing.* Addison-Wesley, Reading, Mass., 1990.
7. Werbos, P. J. *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences,* Ph.D. dissertation. Harvard University, Cambridge, Mass., 1974.
8. Rumelhart, D. E., G. E. Hinton, and R. J. Williams. Learning Internal Representations by Error Propagation. In *Parallel Distributed Processing,* Vol. 1 (D. E. Rumelhart and J. L. McClelland, eds.), MIT Press, Cambridge, Mass., 1986, pp. 318–362.
9. Rohde, G. T. *The Mechanistic Analysis of Pavement Deflections on Subgrades Varying in Stiffness with Depth.* Ph.D. dissertation. Texas A&M University, College Station, 1990.
10. Van Cauwelaert, F. J., D. R. Alexander, T. D. White, and W. R. Barker. Multilayer Elastic Program for Backcalculating Layer Moduli in Pavement Evaluation. In *Nondestructive Testing of Pavements and Backcalculation of Moduli* (A. J. Bush III and G. Y. Baladi, eds.), ASTM 1026, ASTM, Philadelphia, Pa., 1989, pp. 171–188.
11. Hertz, J., A. Krogh, and R. G. Palmer. *Introduction to the Theory of Neural Computation.* Addison-Wesley, Reading, Mass., 1991.
12. American Society for Testing and Materials. Standard Test Method for Deflections with a Falling-Weight-Type Impulse Load Device, In *Annual Book of ASTM Standards,* Vol. 04.03, 1993, pp. 566–568.
13. Matsuoka, K. Noise Injection into Inputs in Back-Propagation Learning. *IEEE Transactions on Systems, Man, and Cybernetics,* Vol. 22, No. 3, 1992, pp. 436–440.
14. Irwin, L., W. Yang, and R. Stubstad. Deflection Reading Accuracy and Layer Thickness Accuracy in Backcalculation of Pavement Layer Moduli. In *Nondestructive Testing of Pavements and Backcalculation of Moduli* (A. J. Bush III and G. Y. Baladi, eds.), ASTM 1026, ASTM, Philadelphia, Pa., 1989, pp. 229–244.
14. Rada, G. R., C. A. Richter, and P. J. Stephanos. Layer Moduli from Deflection Measurements: Software Selection and Development of Strategic Highway Research Program's Procedure for Flexible Pavements. In *Transportation Research Record 1377,* TRB, National Research Council, Washington, D.C., 1992, pp. 77–87.

*Publication of this paper sponsored by Committee on Strength and Deformation Characteristics of Pavement Sections.*