Hindawi Publishing Corporation
ISRN Materials Science
Volume 2013, Article ID 147086, 10 pages
http://dx.doi.org/10.1155/2013/147086

Research Article

Development of a Neural Network Simulator for Studying the Constitutive Behavior of Structural Composite Materials

Hyuntae Na,[1] Seung-Yub Lee,[2] Ersan Üstündag,[2] Sarah L. Ross,[3] Halil Ceylan,[4] and Kasthurirangan Gopalakrishnan[4]

[1] *Department of Computer Science, Iowa State University, Ames, IA 50011, USA*
[2] *Department of Materials Science and Engineering, Iowa State University, Ames, IA 50011, USA*
[3] *Department of Aerospace Engineering, Iowa State University, Ames, IA 50011, USA*
[4] *Department of Civil, Construction and Environmental Engineering, Iowa State University, Ames, IA 50011, USA*

Correspondence should be addressed to Kasthurirangan Gopalakrishnan; rangan@iastate.edu

Received 5 December 2012; Accepted 9 January 2013

Academic Editors: M. Afzaal, F. Ein-Mozaffari, H. Hermann, F. M. Labajos, and H. Yoshihara

This paper introduces a recent development and application of a noncommercial artificial neural network (ANN) simulator with graphical user interface (GUI) to assist in rapid data modeling and analysis in the engineering diffraction field. The real-time network training/simulation monitoring tool has been customized for the study of constitutive behavior of engineering materials, and it has improved data mining and forecasting capabilities of neural networks. This software has been used to train and simulate the finite element modeling (FEM) data for a fiber composite system, both forward and inverse. The forward neural network simulation precisely reduplicates FEM results several orders of magnitude faster than the slow original FEM. The inverse simulation is more challenging; yet, material parameters can be meaningfully determined with the aid of parameter sensitivity information. The simulator GUI also reveals that output node size for materials parameter and input normalization method for strain data are critical train conditions in inverse network. The successful use of ANN modeling and simulator GUI has been validated through engineering neutron diffraction experimental data by determining constitutive laws of the real fiber composite materials via a mathematically rigorous and physically meaningful parameter search process, once the networks are successfully trained from the FEM database.

1. Introduction

Constitutive law (the stress/strain relationship) of the materials determines the mechanical response under quasistatic loading in both elastic and plastic regions and is the most fundamental information for structural applications [1]. While the elastic behavior can be well characterized based on the materials' intrinsic physical properties (E: Young's modulus, G: Shear modulus, ν: Poisson's ratio, etc. (E (or G) and ν are enough for isotropic materials. However, single crystal stiffness (or compliance) tensor needs to be applied for non-isotropic materials, depending on the crystal structure)), the plastic flow is more complicated and depends on the extrinsic material properties (dislocation density, grain size and orientation, available slip systems, etc.) and experimental conditions (temperature, strain rate, sample size and geometry, etc.). Furthermore, since most load bearing materials are not monolithic forms of a single element, obtaining the correct constitutive laws of the composite systems is not a trivial task.

The engineering neutron diffractometer (e.g., SMARTS [2] in Los Alamos National Laboratory (LANL) or VULCAN [3] in Oak Ridge National Laboratory (ORNL)) is an advanced characterization tool that utilizes *in situ* diffraction and loading experiments. Whereas the loading test in the laboratory only gives a macroscopic response from the extensometer or strain gauge, an *in situ* loading/diffraction experiment provides phase-specific, even atomic plane-specific, elastic (lattice) strains from the diffraction patterns as well as macroscopic data (total strain of the sample). The engineering

diffraction field is a research area which characterizes property, condition, and performance of engineering materials through the diffraction technique.

For a more comprehensive interpretation of the neutron diffraction data, mechanics modeling such as self-consistent polycrystalline modeling (SCM) [4] and/or finite element modeling (FEM) has been integrated and proven to be an effective method to define the constitutive behavior of engineering materials. The typical data analysis process is to manually iterate the model fitting onto experimental data by modifying material parameters such as components of the constitutive law. However, these manual fitting processes are often time consuming and inaccurate, even for the experienced scientists especially when the computation time becomes longer and/or the number of fitting parameters is increased. In the process of searching for more rigorous and efficient data analysis tools, artificial neural networks (ANNs) have been found to be a very useful method for engineering diffraction data analysis.

ANN is a valuable computational intelligence system that can simulate the behavior of the human brain and nervous system [5]. An ANN performs two major functions: learning (training) and testing. A specific form of ANN, multilayer perceptrons (MLPs)—also referred to as multilayer feed-forward neural networks—consist of an input layer, one or more hidden (inner) layers, and an output layer. Inside the network, weights are adjusted when data pass between artificial neurons along the connections. For given training data consisting of input-output vectors, values of synaptic weights in an MLP are iteratively updated by a learning algorithm to approximate the target behavior. This process is called "learning" or "training." Training is usually performed by back-propagating (BP) the error signal, layer by layer, and adapting synaptic weights with respect to the magnitude of the error signal [6]. This process is briefly described mathematically as follows.

An artificial neuron receives information (signal) from other neurons, processes it, and then relays the filtered signal to other neurons. The receiving end of the neuron has incoming signals $(x_1, x_2, x_3 \ldots x_n)$. Each of them is assigned a weight (w_{ji}) that is based on experience and likely to change during the training process. The summation of all the weighted signal amounts yields the combined input quantity (I_j) which is sent to a preselected transfer function (f), sometimes called an activation function. A filtered output (y_j) is generated in the outgoing end of the artificial neuron (j) through the mapping of the transfer function. The parameters can be expressed in the form of following equations:

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

$$y_j = f\left(I_j\right). \tag{2}$$

There are several types of transfer functions that can be used, including sigmoid, tangent hyperbolic, threshold, and Gaussian functions. The transfer function most often used is the sigmoid function because of its differentiability. The sigmoid function can be represented by the following equation:

$$f\left(I_j\right) = \frac{1}{1 + \exp\left(-\varphi I_j\right)}, \tag{3}$$

where φ = positive scaling constant, which controls the steepness between the two asymptotic values 0 and 1.

A hyperbolic tangent function (tanh) is also a commonly used (sigmoid) nonlinear activation function for which the amplitude of the output lies in the range $-1 \le f(I_j) \le 1$ and is expressed as follows:

$$f\left(I_j\right) = \frac{\exp\left(\varphi I_j\right) - \exp\left(-\varphi I_j\right)}{\exp\left(\varphi I_j\right) + \exp\left(-\varphi I_j\right)}. \tag{4}$$

In the MLP-BP learning algorithm, the error energy used for monitoring the progress toward convergence is the generalized value of all errors, calculated by a least-squares formulation and represented by a mean-squared error (MSE) as follows:

$$\text{MSE} = \frac{1}{MP}\sum_{1}^{P}\sum_{k=1}^{M}\left(\text{Target}_k - \text{ANN}_k\right)^2. \tag{5}$$

M is the number of neurons in the output layer; P represents the total number of training patterns. Once the training phase of the model has been successfully accomplished, the network performance is verified by presenting independent testing datasets to the ANN. This process is called "testing." The presentation of a complete training set is called an epoch. Additional details regarding the theory and mathematics (including 1 ~ 3) behind ANNs are available in several sources [7–9].

FORTRAN code for the MLP-BP algorithm has been used to train and simulate stress/strain data with reasonable success, in the form of the compiled executable [10]. However, it requires significant efforts in pre- (input files preparation) and post- (visualization) processing, which hinders efficient use of ANN. As a result, the authors have developed the noncommercial ANN simulator graphical user interface in order to (i) monitor training progress of the individual node and/or a series of nodes in real-time for agile training adjustment, (ii) provide an instantaneous visualization tool for both training and simulation, and (iii) optimize the network simulator for study of the constitutive behavior of engineering materials.

In this paper, the authors describe how the software was developed, and how this software contributed to improve neural network training performance and to determine the accurate constitutive laws of the composite materials. The material system used in this study is 40% tungsten-(W-)fiber reinforced bulk-metallic-glass-(BMG-)matrix (Vitreloy 1: $Zr_{41.2}Ti_{13.8}Cu_{12.5}Ni_{10.0}Be_{22.5}$) composites (40 vol% W and 60 vol% BMG). All W-fibers (0.25 mm diameter and 14.4 mm length) are aligned vertically within the cylindrical sample (6 mm diameter and 14.4 mm height) with {110} texture along the loading direction. However, the isotropic nature of tungsten elasticity relieved the extra complication from the

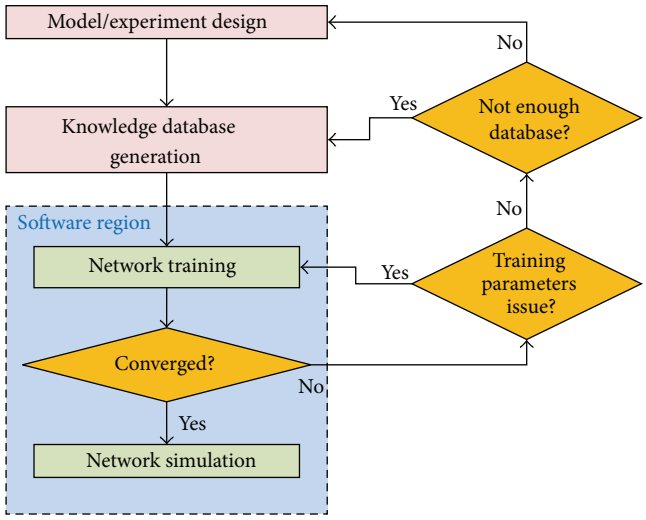


FIGURE 1: Flow chart for the neural network modeling procedure.

crystallographic orientation. Experimental data was obtained from SMARTS engineering neutron diffractometer at Los Alamos. ABAQUS 6.8 was used to build FEM to deduce phase-specific constitutive laws.

2. Software Development

2.1. Work Flow. The general work flow of the neural network training/simulation is illustrated in Figure 1. For successful training, a certain size (database size depends on the complexity of network model. In general, the more, the better, but efficiency should be considered since network training performance saturates above a certain size) knowledge database and meaningful input parameters are necessary regardless of the data origin, whether it is experimental data or a simulation result. Since the training may not be converged, each step needs to be modified based on the feedback as shown in Figure 1. The software presented in this paper was designed to enhance the training and simulation process, highlighted by the dash line enclosure.

In this specific study, there is only one set of compressive loading/diffraction experimental data, whereas 4,500 sets of FEM knowledge database were utilized for network training (elastic strain of tungsten (W) + total strain of the composite (W + BMG) for the same compressive loading sequence). The FEM modeling involved 7 initial input material parameters ($\sigma_0^{\mathrm{W}}, \sigma_1^{\mathrm{W}}, \theta_0^{\mathrm{W}}, \theta_1^{\mathrm{W}}, \sigma_0^{\mathrm{BMG}}, n^{\mathrm{BMG}}$, and T) that define the constitutive behavior of each phase and generated 70 strain outputs for given 35 loading steps. A knowledge database was constructed via *python* script by randomizing the 7 initial parameters within a certain boundary and recording all 7 inputs and 70 outputs in each line. Details on the neuron diffraction experiment can be found in [11], and development of the FEM model and knowledge database is elaborated in [12].

2.2. Software Specification. *Python 2.6* was used as the main programming language aided by a graphic package (*WxPython 2.8*) and optimization modules (*numpy* 1.*3.0* and *scipy 0.7.1*). Although it was designed to provide a convenient environment for the mechanical behavior analysis, the general features as a neural network simulator were not compromised, so that this program can be used for other general purposes. While keeping the FORTRAN engine intact, plotting module, training observer, and optimization module, and so forth, have been added to the main GUI in a modular fashion (as seen in Figure 2, simplified class diagram of the ANN simulator GUI). This software has been tested in Windows and modern Linux systems, and source codes, as well as Windows installer, are available for the public (as of now, the current version is 0.3 released in September, 2011. Software is available upon request; the specific web address is not listed here due to the potential change of release site).

3. Software Application: Training

The ultimate goal of adopting a neural network model in the engineering diffraction field is to determine the best material parameters (constitutive law) from the experimental data with the help of a good mechanics model (FEM) and a rigorous nonlinear mapping algorithm (i.e., ANN). Since the iterative model fitting with experimental data by the input parameter modification is not efficient, ANN can enhance FEM usage in two ways: *forward* and *inverse* neural networks. In the case of forward networks, they try to learn strain output patterns for given material parameters, and inverse networks take the strain data as inputs and correlate them to the proper initial material parameters. Once the forward networks are successfully trained, strain outputs can be obtained within milliseconds compared to the minute long FEM simulation. On the other hand, more powerfully, inverse networks can provide initial material parameters instantaneously without any fitting process if strain data are fed as input vectors. Therefore, there are four use cases in this software: forward training, forward simulation, inverse training, and inverse simulation (these four different use cases were prototyped individually prior to all-in-one ANN GUI development).

3.1. Training Procedure on FEM Data. The MSE value calculated from (5) is what networks are trying to minimize during the training stage for all the output nodes, M, and training data patterns, P. The same principle (MSE) can be applied to the independent testing process with testing data patterns, J, and then (5) can be modified to (6) as a training progress indicator. Since the 4000 and 500 datasets were used for training and testing for each epoch with 70 neurons in the output layer, P, J, and M will be 4000, 500, and 70, respectively, in the current example. Consider

$$\mathrm{MSE} = \frac{1}{MJ}\sum_{1}^{J}\sum_{k=1}^{M}\left(\mathrm{Target}_k - \mathrm{ANN}_k\right)^2. \tag{6}$$

Although (5) and (6) deliver one simple scalar value representing the training and testing performance for each case, it is more practical to monitor testing results by (i)

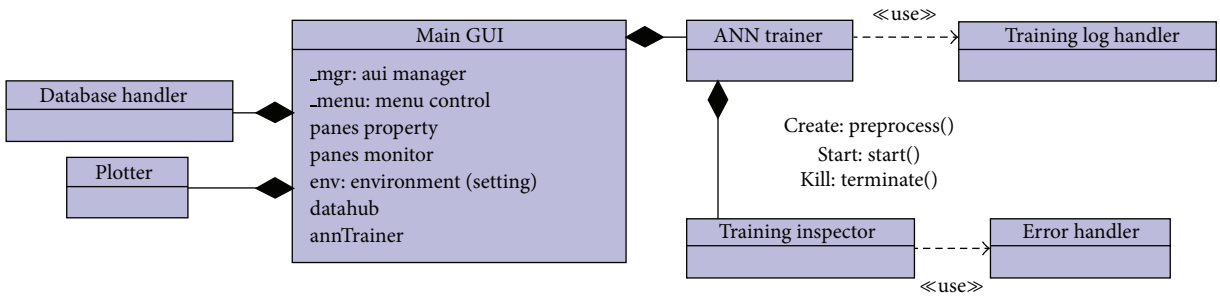


Figure 2: Simplified class diagram of the ANN simulator graphical user interface developed by authors.

the individual node (by default) for the whole testing data, and/or (ii) the whole nodes for each testing set. The modified equation (6) for each node or testing set is shown in (7) or (8), and Figure 3 shows what those equations mean from the data structure of the testing output file. Since the absolute MSE value does not indicate how well ANN and target values are matched, scale normalized residual, R^2, was introduced in (9) and (10). The scaled R^2 varies from 0 (for random correlation) to 1 (perfect correlation), showing how well the training is progressing. Consider

$$\text{MSE (node } m) = \frac{1}{J}\sum_{j=1}^{J}\left(\text{Target}_{(m,j)} - \text{ANN}_{(m,j)}\right)^2, \quad (7)$$

$$\text{MSE (test } j) = \frac{1}{M}\sum_{m=1}^{M}\left(\text{Target}_{(m,j)} - \text{ANN}_{(m,j)}\right)^2, \quad (8)$$

$$R^2 \text{ (node } m) = 1 - \left[\frac{\sum_{j=1}^{J}\left(\text{Target}_{(m,j)} - \text{ANN}_{(m,j)}\right)^2}{\sum_{j=1}^{J}\left(\text{Target}_{(m,j)} - \text{Target}_{(m,\text{avg})}\right)^2}\right], \quad (9)$$

$$R^2 \text{ (test } j) = 1 - \left[\frac{\sum_{m=1}^{M}\left(\text{Target}_{(m,j)} - \text{ANN}_{(m,j)}\right)^2}{\sum_{m=1}^{M}\left(\text{Target}_{(m,j)} - \text{Target}_{(\text{avg},j}\right)^2}\right]. \quad (10)$$

Figure 4 displays ANN GUI with a training in progress. Without this software, one has to process the testing result file (Figure 3) and calculate the residual after the training engine has stopped. Visualizing the full stress/strain profile (by displaying the whole output nodes for each test set) takes even further data processing, and it is practically impossible to check all 500 testing results for every training event. The ANN GUI provides real-time training monitors with various residual functions (MSE, Reduced MSE, Chi square, R^2, and AAE (%) are available in the software). Once the raw database file and training configuration settings are prepared, ANN GUI provides (i) individual node-based monitoring (for entire test data, J, highlighted in blue) and (ii) individual test data-based monitoring (for entire output nodes, M, highlighted in red), according to the selected residual function. The left monitor pane in Figure 4 displays training progress with a selected residual value, and the right pane shows the actual target and ANN values for all 500 (J) testing data for a certain node in real-time. The 45° slope indicates that all data fall into the line of equality, making R^2 approach 1. By scrolling the slide bar, one can monitor all the specific node behavior in the *output node monitor mode* and see the stress/strain plot for all the testing data in the *test data monitor mode*.

3.1.1. Forward Training. One important customization implemented in the ANN GUI is the real-time stress/strain visualization tool as shown in Figure 5. The complete strain profile should be made from the collection of the all the nodes because each node represents an individual strain point for the corresponding loading condition. If the *test data monitor mode* is chosen as a monitoring option, the right pane of Figure 4 displays the stress/strain plot (Figure 5) for an arbitrary test set, j, in real-time. Figure 5 is one of the 500 (J) testing results, corresponding to the median ranked R^2 test set. The network architecture was 7-80-80-70: 7 neurons (materials parameters) in the input layer and 70 neurons in the output layer (70 strain values) with two hidden layers of 80 neurons each. Since this architecture is likely susceptible to result in an overparameterized model with poor generalization performance, the "early stopping" regularization technique was employed during training by periodically checking the performance with a validation set.

3.1.2. Inverse Training. Inverse networks are made by simply swapping the input and output vectors (e.g., *7*-80-80-*70* versus *70*-80-80-*7*), meaning that 70 (old M) strain outputs become input, and the networks try to understand material parameters for a given strain profile. Therefore, in the *output node monitor mode*, the right monitor in Figure 4 can be one of the material parameter training results since the individual node represents one of the 7 (new M) material parameters and should be monitored by each node rather than a series of them.

3.2. Training Parameters

3.2.1. Basic Parameters. Successful training depends on the appropriate selection of training parameters, such as training

	Node 1 (Target)	Node 1 (ANN)	Node m		Node M (Target)	Node M (ANN)
Test 1	Target (1, 1)	ANN (1, 1)	Target (m, 1)	ANN (m, 1)	Target (M, 1)	ANN (M, 1)
	...	...	...	...	...	...
Test j	Target (1, j)	ANN (1, j)	Target (m, j)	ANN (m, j)	Target (M, j)	ANN (M, j)
	...	...	...	...	...	...
Test J	Target (1, J)	ANN (1, J)	Target (m, J)	ANN (m, J)	Target (M, J)	ANN (M, J)

Figure 3: Data structure of testing output file after each training event (epoch). In this case, $J = 500$, and $M = 70$. The actual number of columns is 140 as there are two sets (original target and trained ANN) for each node.

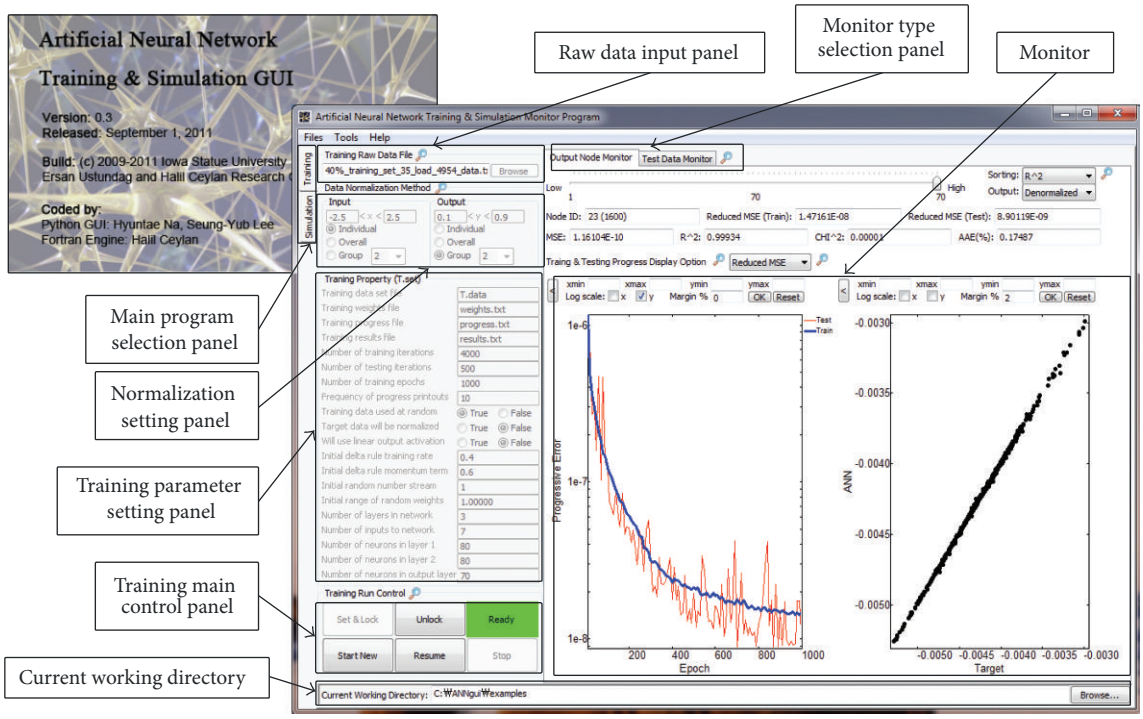


Figure 4: ANN training and simulation GUI outlook. The left monitor displays the training and testing progress via Reduced MSE value (modified from (7)), and the right monitor presents the actual testing results comparing target and ANN values for one of the 70 (M) output nodes in real-time.

time, knowledge database size, learning and momentum rate, network architecture, and data normalization method. Parametric studies that search for optimized training conditions were conducted [13], and their results are listed in Table 1. The ANN GUI facilitated the efficient search process and found that there are basic parameters affecting both the forward and inverse networks as mentioned earlier. However, output layer node size for materials parameters and input layer normalization method for strain data are more critical to inverse network training. Those two parameters are described in the sections that follow in more detail, because they are unique characteristics of the inverse network architecture for the constitutive behavior analysis of composite materials, which were rigorously found through ANN GUI.

3.2.2. Output Layer Node Size for Materials Parameters. It is obvious that the complexity of network training increases with the outer layer (input/output) node size, as the number of neurons within hidden layers is roughly proportional to that of outer layers. Figure 6 shows inverse training results for the σ_0^W (yield strength of the W-fiber) parameter using 500 testing data because the yield strength is the most important component in the constitutive equation. Four different cases are illustrated for comparison, in which the output layer size or the input data normalization method differs. Figures 6(a) and 6(b) show the inverse training results for σ_0^W when all 7 materials parameters are trained together (70-80-80-7 architecture). On the other hand, only the σ_0^W parameter is trained in Figures 6(c) and 6(d) (70-80-80-1 architecture).

Table 1: Optimized forward and inverse ANN training conditions for the case of 7 material parameters and 70 strains (fiber and composite strains).

Epochs	Training data	Network architecture	Learning rate	Momentum rate	Normalization method for strain data
10,000	4,000	7-80-80-70 (Forward) 70-80-80-**1** (Inverse)	0.4	0.6	2-group (Forward) **Individual** (Inverse)

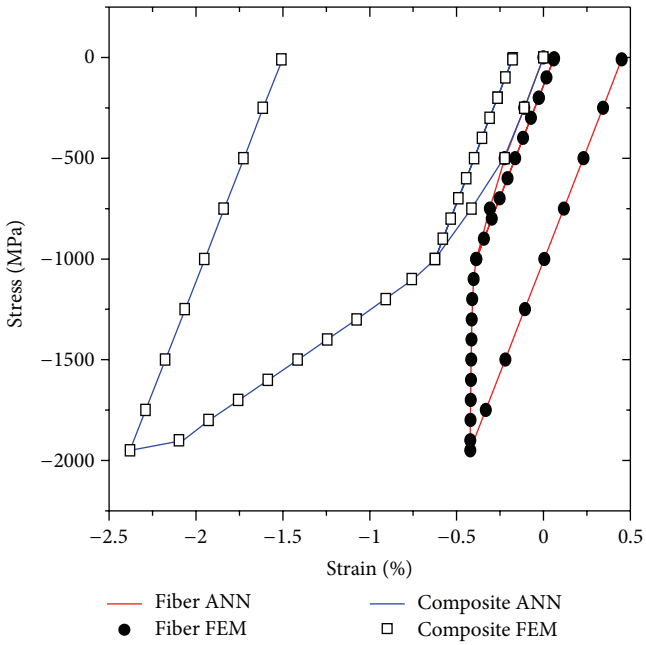


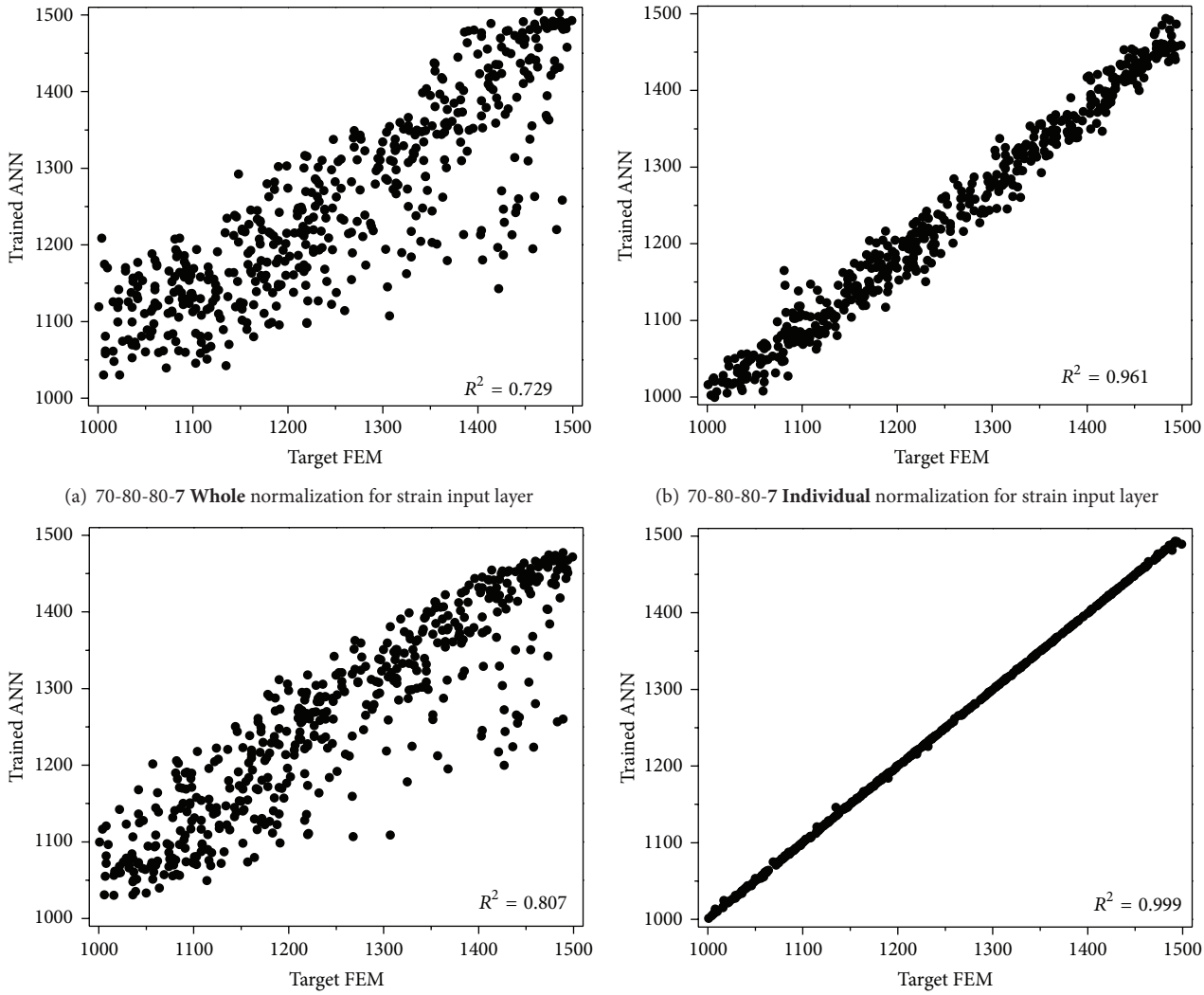
Figure 5: One of the 500 (J) training testing results showing that W-fiber elastic strain and composite total strain are perfectly matched between target FEM and trained ANN. This kind of stress/strain data (for entire output nodes, M) can be directly monitored via the *test data monitor mode* in the GUI. The R^2 values of all the test data are greater than 0.999. The negative sign indicates the compression mode in loading and deformation.

For a given strain input data normalization method, single parameter training results are better than multiparameter training as expected (e.g., compare Figure 6(a) with 6(c), or Figure 6(b) with 6(d)), but this effect was not significant compared to the data normalization effect (e.g., R^2 from 0.729 to 0.807 in Figures 6(a) and 6(c) and R^2 from 0.961 to 0.999 in Figures 6(b) and 6(d)). This observation suggests that material parameters can be trained individually for better inverse analysis results, but the single parameter training does not seem much attractive due to the time-consuming process (by repeating the training 7 times). However, in order to achieve an excellent training result (Figure 6(d), R^2 = 0.999), the single parameter training becomes a necessary condition along with individual group normalization of strain input data. In case of the forward networks, training of the individual strain point is neither sensible nor necessary since it will lose the continuity of the loading process where strains evolve in a progressive manner.

3.2.3. Input Layer Normalization Method for Strain Data. The current network training engine is designed to have an input data range of (−2.5, 2.5) and an output data range of (0.1, 0.9) which requires rescaling of the raw database. In forward training, it is typical to normalize input (material) data (a total of 7 parameters) individually by 7 independent groups, as they are not related to one another. Contrary to the input data, all output data (a total of 70 strains = 2 strain sets × 35 applied stresses) are normalized together as a single group for all the 4,500 datasets because they are basically "one result" of a given set of input parameters (i.e., a self-consistent response of the material to external stimuli). As for the inverse case, the same normalization rules are applied except that the input and output vectors are switched. However, the same normalization (whole normalization for strain data) method was not very effective in inverse training.

For a given output layer, Figures 6(a) and 6(c) show the case of whole strain group (1 group) normalization, whereas Figures 6(b) and 6(d) exhibit the results of the individual group (70 groups) normalization of each strain point in input layers. As seen in Figure 6, training performance is improved a lot by the data normalization method (e.g., R^2 from 0.729 to 0.961 in Figures 6(a) and 6(b) and R^2 from 0.807 to 0.999 in Figures 6(c) and 6(d)). One should note that individual node normalization actually boosts training performance by increasing data resolution at each node, at the expense of data range tolerance or loss of physical meaning. In other words, the whole group normalization is more robust in terms of handling new data which may be out of the original range (e.g., bigger strains in the present case). This point has been confirmed by feeding a new dataset with slightly out-of-range strains throughout the loading/unloading process. Another option is to normalize the strain data into two separate groups, since the fiber elastic strains and total composite strains are physically different entities. In such a case, data tolerance and simulation accuracy are expected to be between the two extreme cases of whole strain group and individual strain node normalizations. In conclusion, the choice of normalization method becomes a matter of compromise between resolution and robustness of material parameters in inverse ANN training, while it is rather insensitive to a series of strain data in the forward case. The ANN GUI provides a convenient, arbitrary data normalization feature for such a training customization.

3.3. Parameter Sensitivity. The effect of the values of individual material parameters on the outcome of a model (often referred to as parameter sensitivity) is crucial information to possess *before* starting an ANN analysis. Knowledge



(a) 70-80-80-7 **Whole** normalization for strain input layer

(b) 70-80-80-7 **Individual** normalization for strain input layer

(c) 70-80-80-**1** **Whole** normalization for strain input layer

(d) 70-80-80-**1** **Individual** normalization for strain input layer

Figure 6: ANN inverse training results for the highly sensitive σ_0^W material parameter (yield strength) from the 500 FEM testing data. Figures 6(a) and 6(b) show the σ_0^W training case in which the other 6 material parameters have also been used. Figures 6(c) and 6(d) illustrate the case in which σ_0^W was trained alone. However, the whole normalization of 70 strains (1 group) over 4,500 datasets generates Figures 6(a) and 6(c), whereas Figures 6(b) and 6(d) come from the individual normalization of the 70 strain input data (70 groups).

on parameter sensitivity is especially vital for successful inverse ANN training. In a previous study [14], parameter sensitivities were determined qualitatively by varying input parameters within their ranges and observing their effects in forward FEM simulations. That study showed that σ_0^W, σ_1^W, and T are the most sensitive parameters, whereas θ_1^W and n^{BMG} exhibit a negligible effect on the simulation results. Table 2 displays the inverse training results using R^2 values for each parameter along with their sensitivity information. It is clear that the success of inverse ANN training for a given material parameter is highly correlated with its sensitivity. This result indicates that the inverse training for an insensitive parameter may still fail, even though all other training conditions are optimum. Table 2 also suggests that one can use the R^2 values of inverse training as a measure of parameter sensitivity. This effect is not significant in the forward case since insensitive materials parameters in the input layer do not influence much the strain simulation results in the output layer.

4. Software Application: Simulation

4.1. Forward Simulation on FEM Data. Once the forward networks are successfully trained, networks with proper weights among interconnected neurons can be used as the alternative simulation engine, which generates strain simulation results several orders of magnitude faster than the original FEM. ANN simulator GUI provides immediate visualization of the stress/strain profile by selecting arbitrary input parameters. This instant response is very efficient for the model fitting

Table 2: Parameter sensitivities qualitatively determined by multiple forward FEM simulations (1st row) and inverse ANN training results (quantified by R^2 from (9), 2nd row) with individual strain data normalization and single parameter training as shown in Figure 6(d) for σ_0^W. Good training results have been obtained for the sensitive parameters, whereas nonsensitive parameters could not be satisfactorily trained. Both qualitative and quantitative sensitivity analyses have been made easy with a real-time monitoring tool of the software.

	σ_0^W	σ_1^W	θ_0^W	θ_1^W	σ_0^{BMG}	n^{BMG}	T
Sensitivity	Highest	Highest	High	Lowest	Low	Lowest	Highest
R^2	0.999	0.999	0.967	0.164	0.405	0.050	0.998

process and can also identify parameter sensitivity both qualitatively (by watching the instantaneous strain profile change based on the input values) and quantitatively (by reading fitting residuals such as R^2 from (10)). Furthermore, the ANN simulator is equipped with an optimization algorithm (Levenberg Marquardt); so, material parameters that generate the least fitting residual between ANN and FEM (or experimental data) can be obtained in a fast and rigorous manner.

4.2. Inverse Simulation on FEM Data. Inverse network simulation finds initial material parameters from the strain profile by inverting the forward network structure. The success of each parameter (node) is represented by R^2 and listed in Table 2. Unlike the forward case, where all R^2 values go beyond 0.999 as seen in Figure 5, only sensitive parameters go above 0.9 in the case of the inverse model. Although it is difficult to estimate the initial parameters precisely, this does not indicate a failure of the training process because it is ascribed to the nature of sensitivity. Once the initial parameters obtained from the inverse network training are put back into the forward model, the ANN simulation against the FEM result mostly reveals the R^2 of 0.99 or better. Therefore, the inverse simulation is highly desirable because one can obtain the initial material parameter immediately without a fitting process, although care must be taken in reading the absolute value because of individual parameter sensitivity.

4.3. Forward/Inverse Simulation on Experimental Data. The highlight of this study with the ANN simulator GUI is finding the most accurate constitutive laws of the W-fiber and BMG-matrix by applying engineering neutron diffraction experimental data to the neural network. Beyond the typical method of iterating model fitting via manual parameter adjustment and judging the fit with the naked eye, ANN offers two additional methods. One way is to obtain phase-specific constitutive laws (4 parameters for the W and 2 parameters for the BMG) plus a freezing temperature (the freezing temperature (T) is a fictive yet important parameter, related to the glass transition temperature of BMG, below which the thermal residual stress between two phases builds up during cool-down process. Although it is not one of the direct parameters for the constitutive laws, it became one of the variables because stress/strain patterns of both phases are affected by this factor on top of their intrinsic constitutive behavior) (T) by feeding experimental data (W-fiber elastic strain and composite total strain) to the FEM trained ANN inverse simulator as inputs. In this case, the inverse simulator suggests the best materials parameters and the freezing temperature without any iteration. The other alternative is to run ANN forward simulation and compare it with experimental strain data until satisfactory match is achieved. This *inverse* (this "*inverse*" does not mean inverse network architecture as mentioned earlier, it indicates the traditional parameter search process since it has to go back (reversely) to the original inputs and changes them to match output data) fitting method with forward simulation seems identical to the traditional one, but it is different in terms of speed and accuracy because the forward neural network engine itself works within milliseconds, unlike FEM, and the parameter optimization module in the GUI provides an automatic search process.

Table 3 summarizes the phase-specific constitutive laws of the 40% W-fiber BMG-matrix composite, obtained from the three methods just described, along with the fitting residual (MSE) for each case: iterative manual fitting of FEM (1st row), inverse neural network (2nd row), and optimizer-assisted forward neural network (3rd row). By comparing MSE values between model and experimental data, manual fitting shows the greatest deviation from each other as expected, while the optimized forward simulation produces the best fitting results. The advantage of the optimizer-assisted forward simulation is that a scientist can interact to judge and feed physical parameters to the networks, and a parameter optimizer provides the best numerical solutions within search boundaries. Figure 7 is the screen shot of the parameter search process by the automatic fitting of the forward ANN model, which is the 3rd case in Table 3. One can produce an instant strain profile for any arbitrary input parameters and then further tune the match between ANN forward and experimental data through the optimizer. In the case of inverse neural network application for the constitutive law, the network architecture and data normalization methods had to be modified from Table 1 (from 70-80-80-1 to 26-50-50-1 architecture and from individual to 2-group normalization). This had to be done to increase the model accuracy and robustness because the current FEM could not reflect reverse yielding behavior during unloading stage, and each experimental strain data for a given stress level may not be necessarily within the range of FEM database as explained in Section 3.2.3.

5. Conclusion

In order to improve the mechanics modeling efficiency and accuracy, artificial neural networks (ANNs) have been

Table 3: Constitutive laws of 40% W-fiber reinforced BMG composite from various approaches. The parentheses in the inverse analysis represent standard deviation from five different runs. During the "forward optimization" process (3rd row), the most insensitive parameter (θ_1^W) was excluded from tuning, and the freezing temperature (T) was also fixed at 355°C because it was found so experimentally from the previous study [15]. Compared to the manual fitting (1st row), the inverse ANN (2nd row) is much more efficient (faster), and forward ANN optimization (3rd row) is both more efficient and accurate.

	σ_0^W (MPa)	σ_1^W (MPa)	θ_0^W (MPa)	θ_1^W (MPa)	σ_0^{BMG} (MPa)	n^{BMG}	T (°C)	MSE
Iterative manual FEM fitting	1305	725	120700	1330	1900	7.3	355	$6.6E-6$
ANN inverse analysis	1234 (7)	729 (1)	566949 (666)	1335 (4)	1898 (1)	10.3 (0.1)	397 (3)	$4.1E-7$
ANN forward optimization	1258	620	200000	1330	1900	8	355	$1.5E-7$

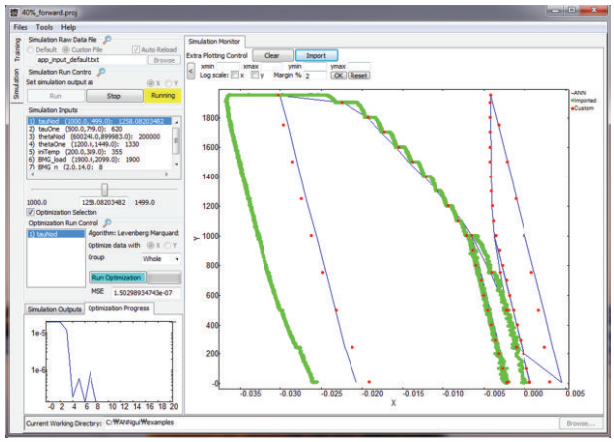


Figure 7: Forward simulation results followed by the optimization process on experimental data. The thick green line is the raw data (total composite strain) from the extensometer, and the red dots are processed experimental data after removing strain relaxation effect from the green line and synchronizing data points with FEM database. The blue line (ANN forward simulation) was fitted by the automatic optimizer to find the best fitting parameter of σ_0^W (yield strength of W-fiber). The observed yield point (~600 MPa) appears much lower than the optimized yield strength (1258 MPa) due to the thermal residual stress effect ("compressive" in W and "tensile" in BMG).

adopted in the engineering diffraction field in this paper. It was demonstrated that ANN can be a faster alternative to the rather slow original model and effective in the parameter search process. However, ANN still required a lot of pre- and postprocessing without any real-time monitoring capability. Therefore, an ANN simulator graphical user interface (GUI), based on the multilayer perceptron back propagation algorithm, was developed and customized for the study of the constitutive behavior of engineering materials. The ANN simulator GUI provides convenient environments for the training/simulation of the forward/inverse models. This software was optimized for the stress/strain analysis, including engineering diffraction data, but its basic features as a neural network simulator have remained intact for general use.

In this specific study involving 40% tungsten-(W-)fiber bulk-metallic-glass-(BMG-)matrix composite, finite element model (FEM) was developed to provide a knowledge database of 4,500 datasets with 7 input material parameters and 70 strain output results for a given geometry and loading condition. The optimum network training conditions on FEM data were found via real-time monitoring for both forward and inverse cases in the "training" mode. Although most training conditions are shared both in forward and inverse models, it was found that the strain input data normalization method and the output materials parameters node size played significant roles in the inverse network model training, which was not readily observed from the previous analyses without ANN GUI. Upon successful training of the networks, the "simulation" mode made it possible to (i) run the ANN simulator more efficiently than FEM from the forward network model and (ii) estimate the initial material parameters of experimental data without any iterative fitting process from the inverse network model. The constitutive laws obtained from both forward and inverse ANN models were better than the manually fitted results, and the automatic parameter optimization algorithm linked to the forward network model provided the most accurate constitutive laws of the composite because of the combination of human interaction (for physically sensible starting/ending points) and rigorous mathematics (for numerically optimized results). The ANN GUI simulator developed in this study can easily be adapted to the study of a number of problems in science, engineering, and business involving input-output nonlinear mapping.

Acknowledgments

This work was partially supported from the DANSE software development project by the NSF Award DMR-0520547. The experimental data has benefited from the use of the Lujan Neutron Scattering Center at LANSCE, which is funded by the Office of Basic Energy Sciences (DOE). Los Alamos National Laboratory is operated by Los Alamos National Security, LLC under DOE Contract DE-AC52-06NA25396.

References

[1] R. Hill, "The essential structure of constitutive laws for metal composites and polycrystals," *Journal of the Mechanics and Physics of Solids*, vol. 15, no. 2, pp. 79–95, 1967.

[2] M. A. M. Bourke, D. C. Dunand, and E. Ustundag, "SMARTS—a spectrometer for strain measurement in engineering materials," *Applied Physics A*, vol. 74, pp. S1707–S1709, 2002.

[3] X. L. Wang, T. M. Holden, G. Q. Rennich et al., "VULCAN—the engineering diffractometer at the SNS," *Physica B*, vol. 385-386, pp. 673–675, 2006.

[4] C. N. Tomé, "Self-consistent polycrystal models: a directional compliance criterion to describe grain interactions," *Modelling and Simulation in Materials Science and Engineering*, vol. 7, no. 5, pp. 723–738, 1999.

[5] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 79, no. 8, pp. 2554–2558, 1982.

[6] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[7] I. Aleksander and H. Morton, *An Introduction to Neural Computing*, Van Nostrand Reinhold Co., New York, NY, USA, 1990.

[8] C. M. Bishop, *Neural Networks for Pattern Recognition*, Oxford University Press, Oxford, UK, 1995.

[9] K. Swingler, *Applying Neural Networks: A Practical Guide*, Academic Press, 1996.

[10] H. Ceylan, *Analysis and Design of Concrete Pavement Systems Using Artificial Neural Networks*, University of Illinois at Urbana-Champaign, Urbana, Ill, USA, 2002.

[11] B. Clausen, S. Y. Lee, E. Üstündag, C. C. Aydiner, R. D. Conner, and M. A. M. Bourke, "Compressive yielding of tungsten fiber reinforced bulk metallic glass composites," *Scripta Materialia*, vol. 49, no. 2, pp. 123–128, 2003.

[12] B. Denizer, E. Ustundag, H. Ceylan, L. Li, and S. Y. Lee, "Engineering neutron diffraction data analysis with inverse neural network modeling," *Materials Science Forum*. In press.

[13] S. L. Ross, "Parametric investigation of artificial neural network development to study composites," unpublished work.

[14] B. Denizer, *Artificial Neural Network Analysis of the Mechanical Properties of Tungsten Fiber/Bulk Metallic Glass Matrix Composites via Neutron Diffraction and Finite Element Modeling*, Iowa State University, Ames, Iowa, USA, 2008.

[15] D. Dragoi, E. Üstündag, B. Clausen, and M. A. M. Bourke, "Investigation of thermal residual stresses in tungsten-fiber/bulk metallic glass matrix composites," *Scripta Materialia*, vol. 45, no. 2, pp. 245–252, 2001.