Artificial Neural Networks for RF and Microwave Design—From Theory to Practice

Qi-Jun Zhang, *Senior Member, IEEE*, Kuldip C. Gupta, *Fellow, IEEE*, and Vijay K. Devabhaktuni, *Student Member, IEEE*

***Abstract*—Neural-network computational modules have recently gained recognition as an unconventional and useful tool for RF and microwave modeling and design. Neural networks can be trained to learn the behavior of passive/active components/circuits. A trained neural network can be used for high-level design, providing fast and accurate answers to the task it has learned. Neural networks are attractive alternatives to conventional methods such as numerical modeling methods, which could be computationally expensive, or analytical methods which could be difficult to obtain for new devices, or empirical modeling solutions whose range and accuracy may be limited. This tutorial describes fundamental concepts in this emerging area aimed at teaching RF/microwave engineers what neural networks are, why they are useful, when they can be used, and how to use them. Neural-network structures and their training methods are described from the RF/microwave designer's perspective. Electromagnetics-based training for passive component models and physics-based training for active device models are illustrated. Circuit design and yield optimization using passive/active neural models are also presented. A multimedia slide presentation along with narrative audio clips is included in the electronic version of this paper. A hyperlink to the *NeuroModeler* demonstration software is provided to allow readers practice neural-network-based design concepts.**

***Index Terms*—Computer-aided design (CAD), design automation, modeling, neural networks, optimization, simulation.**

I. Introduction

NEURAL networks, also called artificial neural networks (ANNs), are information processing systems with their design inspired by the studies of the ability of the human brain to learn from observations and to generalize by abstraction [1]. The fact that neural networks can be trained to learn any arbitrary nonlinear input–output relationships from corresponding data has resulted in their use in a number of areas such as pattern recognition, speech processing, control, biomedical engineering etc. Recently, ANNs have been applied to RF and microwave computer-aided design (CAD) problems as well. Neural networks are first trained to model the electrical behavior of passive and active components/circuits. These trained neural networks, often referred to as neural-network models (or simply neural models), can then be used in high-level simulation and design, providing fast answers to the task they have learned [2], [3]. Neural networks are efficient alternatives to conventional methods such as numerical modeling methods, which could be computationally expensive, or analytical methods, which could be difficult to obtain for new devices, or empirical models, whose range and accuracy could be limited. Neural-network techniques have been used for a wide variety of microwave applications such as embedded passives [4], transmission-line components [5]–[7], vias [8], bends [9], coplanar waveguide (CPW) components [10], spiral inductors [11], FETs [12], amplifiers [13], [14], etc. Neural networks have also been used in impedance matching [15], inverse modeling [16], measurements [17], and synthesis [18].

An increased number of RF/microwave engineers and researchers have started taking serious interest in this emerging technology. As such, this tutorial is prepared to meet the educational needs of the RF/microwave community. The subject of neural networks will be described from the point-of-view of RF/microwave engineers using microwave-oriented language and terminology. In Section II, neural-network structural issues are introduced, and the popularly used multilayer perceptron (MLP) neural network is described at length. Various steps involved in the development of neural-network models are described in Section III. Practical microwave examples illustrating the application of neural-network techniques to component modeling and circuit optimization are presented in Sections IV and V, respectively. Finally, Section VI contains a summary and conclusions. To further aid the readers in quickly grasping the ANN fundamentals and practical aspects, an electronic multimedia slide presentation of the tutorial and a hyperlink to *NeuroModeler* demonstration software[1] are included in the CD-ROM accompanying this issue.

II. Neural-Network Structures

We describe neural-network structural issues to better understand what neural networks are and why they have the ability to represent RF and microwave component behaviors. We study neural networks from the external input–output point-of-view, and also from the internal neuron information

Manuscript received April 25, 2002.

Q.-J. Zhang and V. K. Devabhaktuni are with the Department of Electronics, Carleton University, Ottawa, ON, Canada K1S 5B6 (e-mail: qjz@doe.carleton.ca; vijay@doe.carleton.ca).

K. C. Gupta is with the Department of Electrical and Computer Engineering, University of Colorado at Boulder, Boulder, CO 80309 USA and also with Concept-Modules LLC, Boulder, CO 80303 USA (e-mail: gupta@colorado.edu).

This paper has supplementary downloadable material available at http://ieeexplore.ieee.org, provided by the authors. This includes a Microsoft PowerPoint slide presentation including narrative audio clips and animated transitions. A hyperlink to a Web demonstration of the *NeuroModeler* program is provided in the last slide. This material is 31.7 MB in size.

Digital Object Identifier 10.1109/TMTT.2003.809179

[1] *NeuroModeler*, Q.-J. Zhang, Dept. Electron., Carleton Univ., Ottawa, ON, Canada.

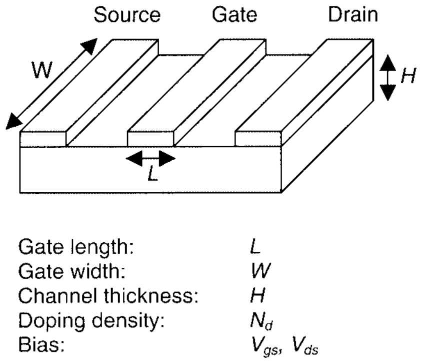


Fig. 1. Physics-based FET to be modeled using a neural network.

processing point-of-view. The most popularly used neural-network structure, i.e., the MLP, is described in detail. The effects of structural issues on modeling accuracy are discussed.

A. Basic Components

A typical neural-network structure has two types of basic components, namely, the processing elements and the interconnections between them. The processing elements are called neurons and the connections between the neurons are known as links or synapses. Every link has a corresponding weight parameter associated with it. Each neuron receives stimulus from other neurons connected to it, processes the information, and produces an output. Neurons that receive stimuli from outside the network are called input neurons, while neurons whose outputs are externally used are called output neurons. Neurons that receive stimuli from other neurons and whose outputs are stimuli for other neurons in the network are known as hidden neurons. Different neural-network structures can be constructed by using different types of neurons and by connecting them differently.

B. Concept of a Neural-Network Model

Let n and m represent the number of input and output neurons of a neural network. Let $\boldsymbol{x}$ be an n-vector containing the external inputs to the neural network, $\boldsymbol{y}$ be an m-vector containing the outputs from the output neurons, and $\boldsymbol{w}$ be a vector containing all the weight parameters representing various interconnections in the neural network. The definition of $\boldsymbol{w}$, and the manner in which $\boldsymbol{y}$ is computed from $\boldsymbol{x}$ and $\boldsymbol{w}$, determine the structure of the neural network.

Consider an FET as shown in Fig. 1. The physical/geometrical/bias parameters of the FET are variables and any change in the values of these parameters affects the electrical responses of the FET (e.g., small-signal S-parameters). Assume that there is a need to develop a neural model that can represent such input–output relationship. Inputs and outputs of the corresponding FET neural model are given by

$$\boldsymbol{x} = [L\ W\ H\ N_{\mathrm{d}}\ V_{\mathrm{gs}}\ V_{\mathrm{ds}}\ \omega]^T \tag{1}$$

$$\boldsymbol{y} = [MS_{11}\ PS_{11}\ MS_{12}\ PS_{12}\ \cdots\ PS_{22}]^T \tag{2}$$

where ω is frequency, and MS_{ij} and PS_{ij} represent magnitude and phase of the S-parameter S_{ij}. The superscript T indicates transpose of a vector or matrix. Other parameters in (1) are defined in Fig. 1. The original physics-based FET modeling problem can be expressed as

$$\boldsymbol{y} = \boldsymbol{f}(\boldsymbol{x}) \tag{3}$$

where $\boldsymbol{f}$ is a detailed physics-based input–output relationship. The neural-network model for the FET is given by

$$\boldsymbol{y} = \boldsymbol{y}(\boldsymbol{x}, \boldsymbol{w}). \tag{4}$$

The neural network in (4) can represent the FET behavior in (3) only after learning the original $\boldsymbol{x}$–$\boldsymbol{y}$ relationship $\boldsymbol{f}$ through a process called training. Several $(\boldsymbol{x}, \boldsymbol{y})$ samples called training data need to be generated either from the FET's physics simulator or from measurements. The objective of training is to adjust neural-network weights $\boldsymbol{w}$ such that the neural model outputs best match the training data outputs. A trained neural model can be used during the microwave design process to provide instant answers to the task it has learned. In the FET case, the neural model can be used to provide fast estimation of S-parameters against the FET's physical/geometrical/bias parameter values.

C. Neural Network Versus Conventional Modeling

The neural-network approach can be compared with conventional approaches for a better understanding. The first approach is the detailed modeling approach (e.g., electromagnetic (EM)-based models for passive components and physics-based models for active devices), where the model is defined by a well-established theory. The detailed models are accurate, but could be computationally expensive. The second approach is an approximate modeling approach, which uses either empirical or equivalent-circuit-based models for passive and active components. These models are developed using a mixture of simplified component theory, heuristic interpretation and representation, and/or fitting of experimental data. Evaluation of approximate models is much faster than that of the detailed models. However, the models are limited in terms of accuracy and input parameter range over which they can be accurate. The neural-network approach is a new type of modeling approach where the model can be developed by learning from detailed (accurate) data of the RF/microwave component. After training, the neural network becomes a fast and accurate model representing the original component behaviors.

D. MLP Neural Network

1) Structure and Notation: MLP is a popularly used neural-network structure. In the MLP neural network, the neurons are grouped into layers. The first and the last layers are called input and output layers, respectively, and the remaining layers are called hidden layers. Typically, an MLP neural network consists of an input layer, one or more hidden layers, and an output layer, as shown in Fig. 2. For example, an MLP neural network with an input layer, one hidden layer, and an output layer, is referred to as three-layer MLP (or MLP3).

Suppose the total number of layers is L. The first layer is the input layer, the Lth layer is the output layer, and layers 2 to $L-1$ are hidden layers. Let the number of neurons in the lth layer be N_l, $l = 1, 2, \ldots, L$. Let w_{ij}^l represent the weight of the link

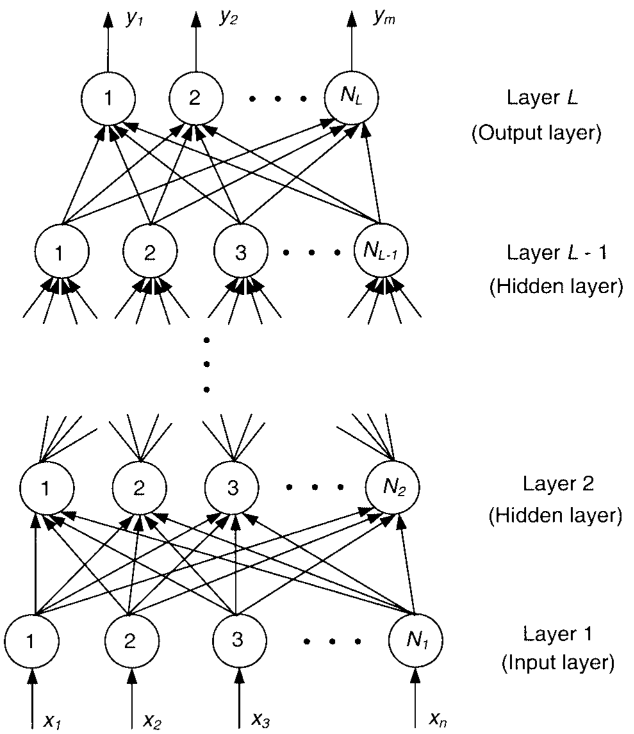


Fig. 2. MLP neural-network structure. Typically, an MLP network consists of an input layer, one or more hidden layers, and an output layer.

between the jth neuron of the $l-1$th layer and the ith neuron of the lth layer. Let x_i represent the ith external input to the MLP and z_i^l be the output of the ith neuron of the lth layer. There is an additional weight parameter for each neuron (w_{i0}^l) representing the bias for the ith neuron of the lth layer. As such, $\boldsymbol{w}$ of the MLP includes w_{ij}^l, $j = 0, 1, \ldots, N_{l-1}$, $i = 1, 2, \ldots, N_l$, and $l = 2, 3, \ldots, L$, i.e., $\boldsymbol{w} = [w_{10}^2\ w_{11}^2\ w_{12}^2\ \cdots\ w_{N_L N_{L-1}}^L]^T$. The parameters in the weight vector are real numbers, which are initialized before MLP training. During training, they are changed (updated) iteratively in a systematic manner [19]. Once the neural-network training is completed, the vector $\boldsymbol{w}$ remains fixed throughout the usage of the neural network as a model.

2) Anatomy of Neurons: In the MLP network, each neuron processes the stimuli (inputs) received from other neurons. The process is done through a function called the activation function in the neuron, and the processed information becomes the output of the neuron. For example, every neuron in the lth layer receives stimuli from the neurons of the $(l-1)$th layer, i.e., z_1^{l-1}, $z_2^{l-1}, \ldots, z_{N_{l-1}}^{l-1}$. A typical ith neuron in the lth layer processes this information in two steps. Firstly, each of the inputs is multiplied by the corresponding weight parameter and the products are added to produce a weighted sum γ_i^l, i.e.,

$$\gamma_i^l = \sum_{j=0}^{N_{l-1}} w_{ij}^l z_j^{l-1}. \tag{5}$$

In order to create the effect of bias parameter w_{i0}^l, we assume a fictitious neuron in the $(l-1)$th layer whose output is $z_0^{l-1} = 1$. Secondly, the weighted sum in (5) is used to activate the neuron's activation function $\sigma(\cdot)$ to produce the final output of the neuron $z_i^l = \sigma(\gamma_i^l)$. This output can, in turn, become the stimulus to neurons in the $(l+1)$th layer. The most commonly used hidden neuron activation function is the sigmoid function given by

$$\sigma(\gamma) = \frac{1}{(1+e^{-\gamma})}. \tag{6}$$

Other functions that can also be used are the arc-tangent function, hyperbolic-tangent function, etc. All these are smooth switch functions that are bounded, continuous, monotonic, and continuously differentiable. Input neurons use a relay activation function and simply relay the external stimuli to the hidden layer neurons, i.e., $z_i^1 = x_i$, $i = 1, 2, \ldots, n$. In the case of neural networks for RF/microwave design, where the purpose is to model continuous electrical parameters, a linear activation function can be used for output neurons. An output neuron computation is given by

$$\sigma(\gamma_i^L) = \gamma_i^L = \sum_{j=0}^{N_{L-1}} w_{ij}^L z_j^{L-1}. \tag{7}$$

3) Feedforward Computation: Given the input vector $\boldsymbol{x} = [x_1\ x_2\ \cdots\ x_n]^T$ and the weight vector $\boldsymbol{w}$, neural network feedforward computation is a process used to compute the output vector $\boldsymbol{y} = [y_1\ y_2\ \cdots\ y_m]^T$. Feedforward computation is useful not only during neural-network training, but also during the usage of the trained neural model. The external inputs are first fed to the input neurons (i.e., first layer) and the outputs from the input neurons are fed to the hidden neurons of the second layer. Continuing this way, the outputs of the $L-1$th layer neurons are fed to the output layer neurons (i.e., the Lth layer). During feedforward computation, neural-network weights $\boldsymbol{w}$ remain fixed. The computation is given by

$$z_i^1 = x_i \quad i = 1, 2, \ldots, N_1 \quad n = N_1 \tag{8}$$

$$z_i^l = \sigma\left(\sum_{j=0}^{N_{l-1}} w_{ij}^l z_j^{l-1}\right), \qquad i = 1, 2, \ldots, N_l; \quad l = 2, 3, \ldots, L \tag{9}$$

$$y_i = z_i^L \quad i = 1, 2, \ldots, N_L \quad m = N_L. \tag{10}$$

4) Important Features: It may be noted that the simple formulas in (8)–(10) are now intended for use as RF/microwave component models. It is evident that these formulas are much easier to compute than numerically solving theoretical EM or physics equations. This is the reason why neural-network models are much faster than detailed numerical models of RF/microwave components. For the FET modeling example described earlier, (8)–(10) will represent the model of S-parameters as functions of transistor gate length, gate width, doping density, and gate and drain voltages. The question of why such simple formulas in the neural network can represent complicated FET (or, in general, EM, physics, RF/microwave) behavior can be answered by the universal approximation theorem.

The universal approximation theorem [20] states that there always exists a three-layer MLP neural network that can approximate any arbitrary nonlinear continuous multidimensional function to any desired accuracy. This forms a theoretical basis for employing neural networks to approximate RF/microwave behaviors, which can be functions of physical/geometrical/bias parameters. MLP neural networks are distributed models, i.e., no single neuron can produce the overall $\boldsymbol{x}$–$\boldsymbol{y}$ relationship. For a given $\boldsymbol{x}$, some neurons are switched on, some are off, and others are in transition. It is this combination of neuron switching states that enables the MLP to represent a given nonlinear input–output mapping. During training process, the MLP's weight parameters are adjusted and, at the end of training, they encode the component information from the corresponding $\boldsymbol{x}$–$\boldsymbol{y}$ training data.

E. Network Size and Layers

For the neural network to be an accurate model of the problem to be learned, a suitable number of hidden neurons are needed. The number of hidden neurons depends upon the degree of nonlinearity of $\boldsymbol{f}$ and the dimensionality of $\boldsymbol{x}$ and $\boldsymbol{y}$ (i.e., values of n and m). Highly nonlinear components need more neurons and smoother items need fewer neurons. However, the universal approximation theorem does not specify as to what should be the size of the MLP network. The precise number of hidden neurons required for a given modeling task remains an open question. Users can use either experience or a trial-and-error process to judge the number of hidden neurons. The appropriate number of neurons can also be determined through adaptive processes, which add/delete neurons during training [4], [21]. The number of layers in the MLP can reflect the degree of hierarchical information in the original modeling problem. In general, the MLPs with one or two hidden layers [22] (i.e., three- or four-layer MLPs) are commonly used for RF/microwave applications.

F. Other Neural-Network Configurations

In addition to the MLP, there are other ANN structures [19], e.g., radial basis function (RBF) networks, wavelet networks, recurrent networks, etc. In order to select a neural-network structure for a given application, one starts by identifying the nature of the $\boldsymbol{x}$–$\boldsymbol{y}$ relationship. Nondynamic modeling problems (or problems converted from dynamic to nondynamic using methods like harmonic balance) can be solved using MLP, RBF, and wavelet networks. The most popular choice is the MLP since its structure and training are well-established. RBF and wavelet networks can be used when the problem exhibits highly nonlinear and localized phenomena (e.g., sharp variations). Time-domain dynamic responses such as those in nonlinear modeling can be represented using recurrent neural networks [13] and dynamic neural networks [14]. One of the most recent research directions in the area of microwave-oriented ANN structures is the knowledge-based networks [6]–[9], which combine existing engineering knowledge (e.g., empirical equations and equivalent-circuit models) with neural networks.

III. Neural-Network Model Development

The neural network does not represent any RF/microwave component unless we train it with RF/microwave data. To develop a neural-network model, we need to identify input and output parameters of the component in order to generate and preprocess data, and then use this data to carry out ANN training. We also need to establish quality measures of neural models. In this section, we describe the important steps and issues in neural model development.

A. Problem Formulation and Data Processing

1) ANN Inputs and Outputs: The first step toward developing a neural model is the identification of inputs ($\boldsymbol{x}$) and outputs ($\boldsymbol{y}$). The output parameters are determined based on the purpose of the neural-network model. For example, real and imaginary parts of S-parameters can be selected for passive component models, currents and charges can be used for large-signal device models, and cross-sectional resistance–inductance–conductance–capacitance (RLGC) parameters can be chosen for very large scale integration (VLSI) interconnect models. Other factors influencing the choice of outputs are: 1) ease of data generation; 2) ease of incorporation of the neural model into circuit simulators, etc. Neural model input parameters are those device/circuit parameters (e.g., geometrical, physical, bias, frequency, etc.) that affect the output parameter values.

2) Data Range and Sample Distribution: The next step is to define the range of data to be used in ANN model development and the distribution of $\boldsymbol{x}$–$\boldsymbol{y}$ samples within that range. Suppose the range of input space (i.e., $\boldsymbol{x}$-space) in which the neural model would be used after training (during design) is $[\boldsymbol{x}_{\min}, \boldsymbol{x}_{\max}]$. Training data is sampled slightly beyond the model utilization range, i.e., $[\boldsymbol{x}_{\min} - \Delta, \boldsymbol{x}_{\max} + \Delta]$, in order to ensure reliability of the neural model at the boundaries of model utilization range. Test data is generated in the range $[\boldsymbol{x}_{\min}, \boldsymbol{x}_{\max}]$.

Once the range of input parameters is finalized, a sampling distribution needs to be chosen. Commonly used sample distributions include uniform grid distribution, nonuniform grid distribution, design of experiments (DOE) methodology [8], star distribution [9], and random distribution. In uniform grid distribution, each input parameter x_i is sampled at equal intervals. Suppose the number of grids along input dimension x_i is n_i. The total number of $\boldsymbol{x}$–$\boldsymbol{y}$ samples is given by $P = \prod_{i=1}^{n} n_i$. For example, in an FET modeling problem where $\boldsymbol{x} = [V_{\rm gs}\ V_{\rm ds}\ {\rm freq}]^T$ and neural model utilization range is

$$\begin{bmatrix} -5\ \text{V} \\ 0\ \text{V} \\ 1\ \text{GHz} \end{bmatrix} \leq \boldsymbol{x} \leq \begin{bmatrix} 0\ \text{V} \\ 10\ \text{V} \\ 20\ \text{GHz} \end{bmatrix} \tag{11}$$

training data can be generated in the range

$$\begin{bmatrix} -5 - 0.5 \\ 0 \\ 1 - 0.5 \end{bmatrix} \leq \boldsymbol{x} \leq \begin{bmatrix} 0 \\ 10 + 1 \\ 20 + 2 \end{bmatrix}. \tag{12}$$

In nonuniform grid distribution, each input parameter is sampled at unequal intervals. This is useful when the problem behavior is highly nonlinear in certain subregions of the $\boldsymbol{x}$-space and dense sampling is needed in such subregions. Modeling dc characteristics (I–V curves) of an FET is a classic example for nonuniform grid distribution. Sample distributions based on DOE (e.g., 2^n factorial experimental design, central composite experimental design) and star distribution are used in situations where training data generation is expensive.

3) Data Generation: In this step, $\boldsymbol{x}$–$\boldsymbol{y}$ sample pairs are generated using either simulation software (e.g., three-dimensional (3-D) EM simulations using Ansoft *HFSS*[2]) or measurement setup (e.g., S-parameter measurements from a *network analyzer*). The generated data could be used for training the neural network and testing the resulting neural-network model. In practice, both simulations and measurements could have small errors. While errors in simulation could be due to truncation/roundoff or nonconvergence, errors in measurement could be due to equipment limitations or tolerances. Considering this, we introduce a vector $\boldsymbol{d}$ to represent the outputs from simulation/measurement corresponding to an input $\boldsymbol{x}$. Data generation is then defined as the use of simulation/measurement to obtain sample pairs ($\boldsymbol{x}_k$, $\boldsymbol{d}_k$), $k = 1, 2, \ldots, P$. The total number of samples P is chosen such that the developed neural model best represents the given problem $\boldsymbol{f}$. A general guideline is to generate larger number of samples for a nonlinear high-dimensional problem and fewer samples for a relatively smooth low-dimensional problem.

4) Data Organization: The generated ($\boldsymbol{x}$, $\boldsymbol{d}$) sample pairs could be divided into three sets, namely, training data, validation data, and test data. Let T_r, V, T_e, and D represent index sets of training data, validation data, test data, and generated (available) data, respectively. Training data is utilized to guide the training process, i.e., to update the neural-network weight parameters during training. Validation data is used to monitor the quality of the neural-network model during training and to determine stop criteria for the training process. Test data is used to independently examine the final quality of the trained neural model in terms of accuracy and generalization capability.

Ideally, each of the data sets T_r, V, and T_e should adequately represent the original component behavior $\boldsymbol{y} = \boldsymbol{f}(\boldsymbol{x})$. In practice, available data D can be split depending upon its quantity. When D is sufficiently large, it can be split into three mutually disjoint sets. When D is limited due to expensive simulation or measurement, it can be split into just two sets. One of the sets is used for training and validation $(T_r = V)$ and the other for testing (T_e) or, alternatively, one of the sets is used for training (T_r) and the other for validation and testing $(V = T_e)$.

5) Data Preprocessing: Contrary to binary data (0's and 1's) in pattern recognition applications, the orders of magnitude of various input ($\boldsymbol{x}$) and output ($\boldsymbol{d}$) parameter values in microwave applications can be very different from one another. As such, a systematic preprocessing of training data called scaling is desirable for efficient neural-network training. Let x, $x_{\min}$, and $x_{\max}$ represent a generic input element in the vectors $\boldsymbol{x}$, $\boldsymbol{x}_{\min}$, and $\boldsymbol{x}_{\max}$ of original (generated) data, respectively. Let $\tilde{x}$, $\tilde{x}_{\min}$, and $\tilde{x}_{\max}$ represent a generic element in the vectors $\tilde{\boldsymbol{x}}$, $\tilde{\boldsymbol{x}}_{\min}$, and $\tilde{\boldsymbol{x}}_{\max}$ of scaled data, where $[\tilde{\boldsymbol{x}}_{\min}, \tilde{\boldsymbol{x}}_{\max}]$ is the input parameter range after scaling. Linear scaling is given by

$$\tilde{x} = \tilde{x}_{\min} + \frac{x - x_{\min}}{x_{\max} - x_{\min}} (\tilde{x}_{\max} - \tilde{x}_{\min}) \tag{13}$$

and corresponding de-scaling is given by

$$x = x_{\min} + \frac{\tilde{x} - \tilde{x}_{\min}}{\tilde{x}_{\max} - \tilde{x}_{\min}} (x_{\max} - x_{\min}). \tag{14}$$

[2]*HFSS, ver. 8.0*, Ansoft Corporation, Pittsburgh, PA.

Output parameters in training data, i.e., elements in $\boldsymbol{d}$, can also be scaled in a similar manner. Linear scaling of data can provide balance between different inputs (or outputs) whose values are different by orders of magnitude. Another scaling method is the logarithmic scaling [1], which can be applied to outputs with large variations in order to provide a balance between small and large values of the same output. In the training of knowledge-based networks, where knowledge neuron functions (e.g., Ohm's law, Faraday's law, etc.) require preservation of physical meaning of the input parameters, training data is not scaled, i.e., $x = \tilde{x}$. At the end of this step, the scaled data is ready to be used for training.

B. Neural-Network Training

1) Weight Parameters Initialization: In this step, we prepare the neural network for training. The neural-network weight parameters ($\boldsymbol{w}$) are initialized so as to provide a good starting point for training (optimization). The widely used strategy for MLP weight initialization is to initialize the weights with small random values (e.g., in the range [−0.5, 0.5]). Another method suggests that the range of random weights be inversely proportional to the square root of number of stimuli a neuron receives on average. To improve the convergence of training, one can use a variety of distributions (e.g., Gaussian distribution) and/or different ranges and different variances for the random number generators used in initializing the ANN weights [23].

2) Formulation of Training Process: The most important step in neural model development is the neural-network training. The training data consists of sample pairs $\{(\boldsymbol{x}_k, \boldsymbol{d}_k)$, and $k \in T_r\}$, where $\boldsymbol{x}_k$ and $\boldsymbol{d}_k$ are n- and m-vectors representing the inputs and desired outputs of the neural network. We define neural-network training error as

$$E_{T_r}(\boldsymbol{w}) = \frac{1}{2} \sum_{k \in T_r} \sum_{j=1}^{m} \left| y_j(\boldsymbol{x}_k, \boldsymbol{w}) - d_{jk} \right|^2 \tag{15}$$

where d_{jk} is the jth element of $\boldsymbol{d}_k$ and $y_j(\boldsymbol{x}_k, \boldsymbol{w})$ is the jth neural-network output for input $\boldsymbol{x}_k$.

The purpose of neural-network training, in basic terms, is to adjust $\boldsymbol{w}$ such that the error function $E_{T_r}(\boldsymbol{w})$ is minimized. Since $E_{T_r}(\boldsymbol{w})$ is a nonlinear function of the adjustable (i.e., trainable) weight parameters $\boldsymbol{w}$, iterative algorithms are often used to explore the $\boldsymbol{w}$-space efficiently. One begins with an initialized value of $\boldsymbol{w}$ and then iteratively updates it. Gradient-based iterative training techniques update $\boldsymbol{w}$ based on error information $E_{T_r}(\boldsymbol{w})$ and error derivative information $\partial E_{T_r}/\partial \boldsymbol{w}$. The subsequent point in $\boldsymbol{w}$-space denoted as $\boldsymbol{w}_{\text{next}}$ is determined by a step down from the current point $\boldsymbol{w}_{\text{now}}$ along a direction vector $\boldsymbol{h}$, i.e., $\boldsymbol{w}_{\text{next}} = \boldsymbol{w}_{\text{now}} + \eta \boldsymbol{h}$. Here, $\Delta \boldsymbol{w} = \eta \boldsymbol{h}$ is called the weight update and η is a positive step size known as the learning rate. For example, the backpropagation (BP) training algorithm [19] updates $\boldsymbol{w}$ along the negative direction of the gradient of training error as $\boldsymbol{w} = \boldsymbol{w} - \eta (\partial E_{T_r}/\partial \boldsymbol{w})$.

3) Error Derivative Computation: As mentioned earlier, gradient-based training techniques require error derivative computation, i.e., $\partial E_{T_r}/\partial \boldsymbol{w}$. For the MLP neural network, these derivatives are computed using a standard approach often

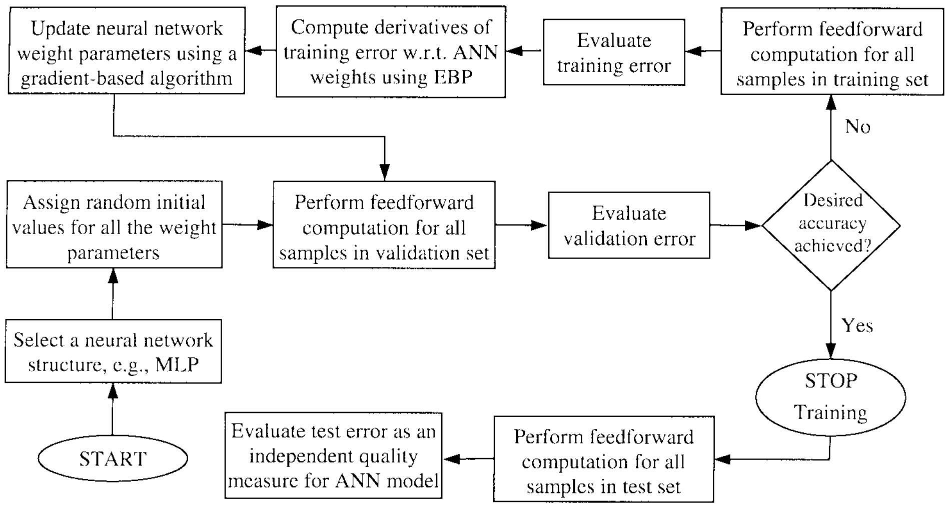


Fig. 3. Flowchart demonstrating neural-network training, neural model testing, and use of training, validation, and test data sets in ANN modeling approach.

referred to as error backpropagation (EBP), which is described here. We define a per-sample error function E_k given by

$$E_k = \frac{1}{2}\sum_{j=1}^{m}\left(y_j(\boldsymbol{x}_k, \boldsymbol{w}) - d_{jk}\right)^2 \tag{16}$$

for the kth data sample $k \in T_r$. Let δ_i^L represent the error between the ith neural-network output and the ith output in training data, i.e.,

$$\delta_i^L = y_i(\boldsymbol{x}_k, \boldsymbol{w}) - d_{ik}. \tag{17}$$

Starting from the output layer, this error can be backpropagated to the hidden layers as

$$\delta_i^l = \left(\sum_{j=1}^{N_{l+1}} \delta_j^{l+1} w_{ji}^{l+1}\right) z_i^l(1 - z_i^l), \quad l = L-1, L-2, \ldots, 3, 2 \tag{18}$$

where δ_i^l represents the local error at the ith neuron in the lth layer. The derivative of the per-sample error in (16) with respect to a given neural-network weight parameter w_{ij}^l is given by

$$\frac{\partial E_k}{\partial w_{ij}^l} = \delta_i^l z_j^{l-1} \qquad l = L, L-1, \ldots, 2. \tag{19}$$

Finally, the derivative of the training error in (15) with respect to w_{ij}^l can be computed as

$$\frac{\partial E_{T_r}}{\partial w_{ij}^l} = \sum_{k \in T_r} \frac{\partial E_k}{\partial w_{ij}^l}.$$

Using EBP, $(\partial E_{T_r}/\partial \boldsymbol{w})$ can be systematically evaluated for the MLP neural-network structure and can be provided to gradient-based training algorithms for the determination of weight update $\Delta \boldsymbol{w}$.

4) More About Training: Validation error E_V and test error E_{T_e} can be defined in a manner similar to (15) using the validation and test data sets V and T_e. During ANN training, validation error is periodically evaluated and the training is terminated once a reasonable E_V is reached. At the end of the training, the quality of the neural-network model can be independently assessed by evaluating the test error E_{T_e}. Neural-network training algorithms commonly used in RF/microwave applications include gradient-based training techniques such as BP, conjugate-gradient, quasi-Newton, etc. Global optimization methods such as simulated annealing and genetic algorithms can be used for globally optimal solutions of neural-network weights. However, the training time required for global optimization methods is much longer than that for gradient-based training techniques.

The neural-network training process can be categorized into sample-by-sample training and batch-mode training. In sample-by-sample training, also called online training, $\boldsymbol{w}$ is updated each time a training sample $(\boldsymbol{x}_k, \boldsymbol{d}_k)$ is presented to the network. In batch-mode training, also known as offline training, $\boldsymbol{w}$ is updated after each epoch, where an epoch is defined as a stage of the training process that involves presentation of all the training data (or samples) to the neural network once. In the RF/microwave case, batch-mode training is usually more effective.

A flowchart summarizing major steps in neural-network training and testing is shown in Fig. 3.

5) Over-Learning and Under-Learning: The ability of a neural network to estimate output $\boldsymbol{y}_k$ accurately when presented with input $\boldsymbol{x}_k$ never seen during training (i.e., $k \notin T_r$) is called generalization ability. The normalized training error is defined as

$$\hat{E}_{T_r}(\boldsymbol{w}) = \left[\frac{1}{mP_{T_r}} \sum_{k \in T_r} \sum_{j=1}^{m} \left|\frac{y_j(\boldsymbol{x}_k, \boldsymbol{w}) - d_{jk}}{d_{\max, j} - d_{\min, j}}\right|^2\right]^{1/2} \tag{20}$$

where $d_{\min, j}$ and $d_{\max, j}$ are the minimum and maximum values of the jth element of all $\boldsymbol{d}_k$, $k \in D$, and P_{T_r} is the number of data samples in T_r. The normalized validation error $\hat{E}_V$ can be similarly defined. *Good learning* of a neural network is achieved when both $\hat{E}_{T_r}$ and $\hat{E}_V$ have small values (e.g., 0.50%) and are close to each other. The ANN exhibits *over-learning* when it memorizes the training data, but cannot generalize well (i.e., $\hat{E}_{T_r}$ is small, but $\hat{E}_V \gg \hat{E}_{T_r}$). Remedies for over-learning are: 1) deleting a certain number of hidden neurons or 2) adding more samples to the training data. The neural network exhibits *under-learning*, when it has difficulties in learning the training data itself (i.e., $\hat{E}_{T_r} \gg 0$). Possible remedies are: 1) adding more hidden neurons or 2) perturbing the current solution $\boldsymbol{w}$ to escape from a local minimum of $E_{T_r}(\boldsymbol{w})$, and then continuing training.

6) Quality Measures: The quality of a trained neural-network model is evaluated with an independent set of data, i.e., T_e. We define a relative error δ_{jk} for the jth output of the neural model for the kth test sample as

$$\delta_{jk} = \frac{y_j(\boldsymbol{x}_k, \boldsymbol{w}) - d_{jk}}{d_{\max, j} - d_{\min, j}}, \qquad j = 1, \ldots, m, k \in T_e. \quad (21)$$

A quality measure based on the qth norm is then defined as

$$M_q = \left[\sum_{k \in T_e} \sum_{j=1}^{m} |\delta_{jk}|^q \right]^{1/q}. \quad (22)$$

The average test error can be calculated using M_1 as

$$\text{Average Test Error} = \frac{M_1}{m P_{T_e}} \quad (23)$$

where P_{T_e} represents number of samples in test set T_e. The worst case error among all test samples and all neural-network model outputs can be calculated using

$$M_\infty = \max_{k \in T_e} \left(\max_{j=1}^{m} |\delta_{jk}| \right). \quad (24)$$

Other statistical measures such as correlation coefficient and standard deviation can also be used.

IV. Component Modeling Using Neural Networks

Component/device modeling is one of the most important areas of RF/microwave CAD. The efficiency of CAD tools depends largely on speed and accuracy of the component models. Development of neural-network models for active devices, passive components, and high-speed interconnects has already been demonstrated [6], [8], [24]. These neural models could be used in device level analysis and also in circuit/system-level design [10], [12]. In this section, neural-network modeling examples are presented in each of the above-mentioned categories.

A. High-Speed Interconnect Network

In this example, a neural network was trained to model signal propagation delays of a VLSI interconnect network in printed circuit boards (PCBs). The electrical equivalent circuit showing the interconnection of a source integrated circuit (IC) pin to the receiver pins is shown in Fig. 4. During PCB design, each individual interconnect network needs to be varied in terms of its interconnect lengths, receiver-pin load characteristics, source characteristics, and network topology. To facilitate this, a neural-network model of the interconnect configuration was developed [24].

The input variables in the model are $\boldsymbol{x} = [L_i\, R_i\, C_i\, R_s\, V_p\, T_r\, e_j]^T$, $i = 1, 2, 3, 4$ and $j = 1, 2, 3$. Here, L_i is length of the ith interconnect, R_i and C_i are terminations of the ith interconnect, R_s is the source impedance, and V_p and T_r are peak value and rise time of the source voltage. The parameter e_j identifies the interconnect network topology [1], [24], as defined in Fig. 5. The outputs of the neural model are the propagation delays at the four terminations, i.e., $\boldsymbol{y} = [\tau_1\, \tau_2\, \tau_3\, \tau_4]^T$, where propagation delay is defined as the time taken for the signal to reach 80% of its steady-state value. The corresponding three-layer MLP neural-network structure, shown in Fig. 6, has a total of 18 input neurons and four output neurons.

Due to large dimensionality of the input parameters (i.e., $n = 18$), a relatively large number of training and test data (4500 + 4500) and 40 hidden layer neurons were used. Data was generated using an in-house interconnect simulator NILT [25]. Neural-network training was carried out using a classical BP algorithm for MLPs. The average test error of the resulting neural network model was observed to be 0.04%. This fast and accurate neural-network model was used in PCB interconnect

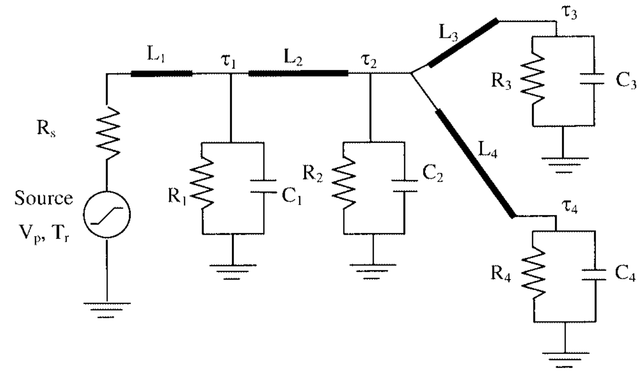


Fig. 4. Circuit representation of the VLSI interconnect network showing the connection of a source IC pin to four receiver pins. A neural model is to be developed to represent the signal delays at the four receiver pins as functions of the interconnect network parameters.

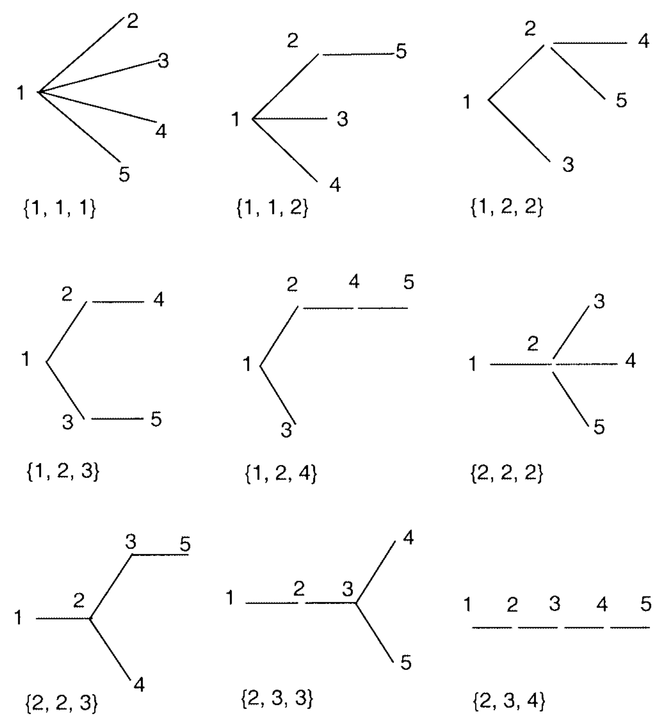


Fig. 5. Possible network configurations for four interconnect lines in a tree interconnect network. The values of the neural-network input variables $\{e_1\, e_2\, e_3\}$ are shown in curly brackets. Each combination of these input variables defines a particular interconnect topology [1], [24].

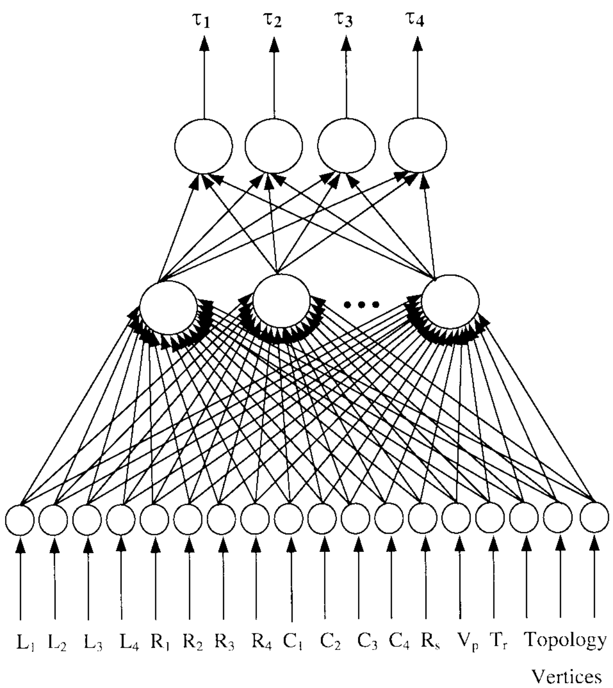


Fig. 6. Three-layer MLP structure with 18 inputs and four outputs used for modeling the interconnect network of Fig. 4.

simulation, where 20 000 interconnect trees (with different interconnect lengths, terminations, and topologies) had to be repetitively analyzed. Neural-model-based simulation was observed to be 310 times faster than existing NILT interconnect network simulator. This enhanced model efficiency becomes important for the design of large VLSI circuits.

B. CPW Symmetric T-Junction

At microwave and millimeter-wave frequencies, CPW circuits offer several advantages, such as the ease of shunt and series connections, low radiation, low dispersion, and avoidance of the need for thin fragile substrates. Currently, CAD tools available for CPW circuits are inadequate because of the nonavailability of fast and accurate models for CPW discontinuities such as T-junctions, bends, etc. Much effort has been expended in developing efficient methods for EM simulation of CPW discontinuities. However, the time-consuming nature of EM simulations limits the use of these tools for interactive CAD, where the geometry of the component needs to be repetitively changed, thereby necessitating massive EM simulations. Neural-network-based modeling and CAD approach addresses this challenge.

In this example, the neural-network model of a symmetric T-junction [10] is described. The T-junction configuration is similar to that of the 2T junction shown in Fig. 7. Variable neural model input parameters are the physical dimensions $W_{\rm in}$, $G_{\rm in}$, $W_{\rm out}$, and $G_{\rm out}$, and the frequency of operation, i.e., $\boldsymbol{x} = [W_{\rm in}\, G_{\rm in}\, W_{\rm out}\, G_{\rm out}\, {\rm freq}]^T$. Parameters $W_{\rm in}$ and $G_{\rm in}$ are strip width and gap dimension of the side CPW, where $W_{\rm out}$ and $G_{\rm out}$ specify the two colinear CPW lines. Air bridges shown in T-junctions of Fig. 7 are also included for the model development. The outputs of the ANN model are the magnitudes and phases of the S-parameters, i.e., $\boldsymbol{y} = [MS_{11}\ PS_{11}\, MS_{13}\, PS_{13}\ MS_{23}\, PS_{23}\, MS_{33}\, PS_{33}]^T$.

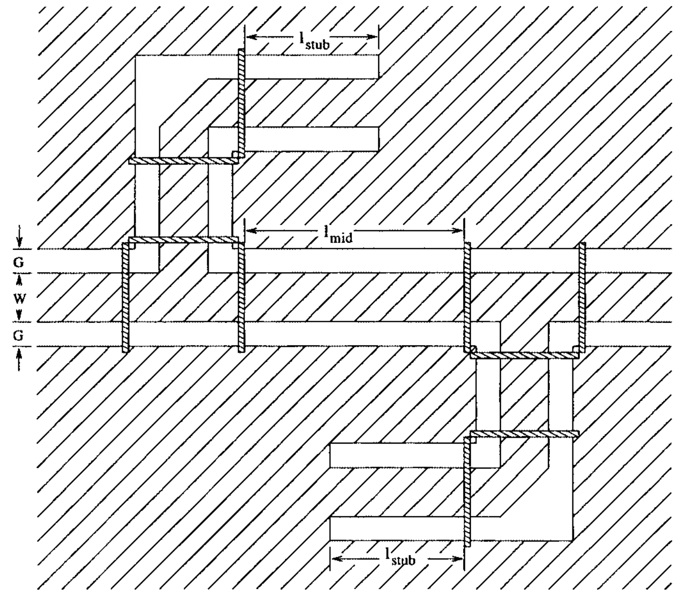


Fig. 7. Geometry of the CPW folded double-stub filter circuit. Optimization of the filter is carried out using fast EM-ANN models of various components in the circuit.

TABLE I
ERROR COMPARISON BETWEEN THE ANN MODEL AND EM SIMULATIONS FOR THE CPW SYMMETRIC T-JUNCTION. INPUT BRANCHLINE PORT IS PORT 1 AND THE OUTPUT PORTS ON THE MAIN LINE ARE PORTS 2 AND 3

	$\lvert S_{11}\rvert$	$\angle S_{11}$	$\lvert S_{13}\rvert$	$\angle S_{13}$	$\lvert S_{23}\rvert$	$\angle S_{23}$	$\lvert S_{33}\rvert$	$\angle S_{33}$
Training								
Average Error	0.00150	0.754	0.00071	0.176	0.00084	0.246	0.00106	0.633
Std. Deviation	0.00128	0.696	0.00058	0.172	0.00097	0.237	0.00109	0.546
Testing								
Average Error	0.00345	0.782	0.00088	0.141	0.00126	0.177	0.00083	0.838
Std. Deviation	0.00337	0.674	0.00085	0.125	0.00105	0.129	0.00068	0.717

Data generation was performed for 25 physical configurations (sizes) of the component over a frequency range of 1–50 GHz. The generated data was split into 155 training and 131 test samples. A three-layer MLP neural-network structure with 15 hidden layer neurons was trained using the BP algorithm. The accuracy of the developed neural models is shown in Table I in terms of average error and standard deviation.

C. Transistor Modeling

Physics-based device models are CPU intensive, especially when used for high-level design involving repetitive simulations. A neural-network model will be very efficient for this kind of devices in speeding up simulation and optimization. Two illustrative examples are presented for neural-network transistor models.

In the first example, a neural-network model representing large-signal behavior of a MESFET was developed [12]. The neural-network model has six inputs, i.e., gate length (L), gatewidth (W), channel thickness (a), doping density $(N_{\rm d})$, gate–source voltage $(V_{\rm gs})$, and drain–source voltage $(V_{\rm ds})$. Under normal operating conditions, the gate is reverse biased and gate conduction current $(i_{\rm gc})$ can be neglected. As a result,

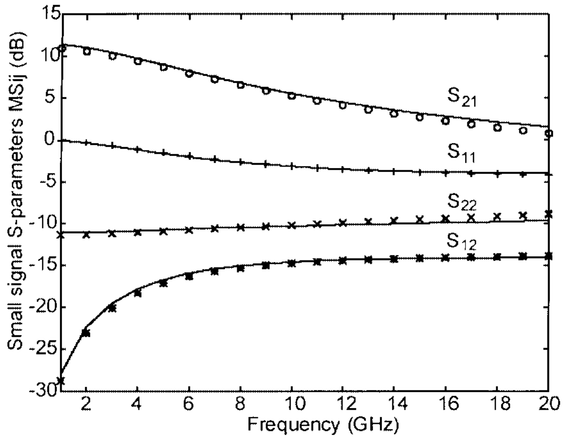


Fig. 8. Comparison of small-signal S-parameter predictions from the large-signal MESFET neural-network model (o, $+$, x, $*$) with those from the Khatibzadeh and Trew MESFET model (—).

drain and source conduction currents $i_{\rm dc}$ and $i_{\rm sc}$ are equal. The neural-network model has four outputs including the drain current and electrode charges, i.e., $\boldsymbol{y} = [i_{\rm d}\, q_{\rm g}\, q_{\rm d}\, q_s]^T$. A three-layer MLP neural-network structure was used. Training and test data (a total of 1000 samples) were generated from *OSA90*[3] simulations using a semianalytical MESFET model by Khatibzadeh and Trew [26]. The neural network was trained using a modified BP algorithm including momentum adaptation to improve the speed of convergence. The trained neural model accurately predicted dc/ac characteristics of the MESFET. A comparison of the MESFET neural model's S-parameter predictions versus those from the Khatibzadeh and Trew MESFET model is shown in Fig. 8. Since the neural model directly describes terminal currents and charges as nonlinear functions of device parameters, it can be conveniently used for harmonic-balance simulations.

In the second example, neural-network models representing dc characteristics of a MOSFET were developed based on physics-based data obtained by using a recent automatic model generation algorithm [27]. The neural-network model has two inputs, i.e., drain voltage $(V_{\rm d})$ and gate voltage $(V_{\rm g})$. Drain current $(I_{\rm d})$ is the neural model output parameter. Training and test data were generated using a physics-based *MINIMOS* simulator.[4] The average test errors of the trained MOSFET neural models were observed to be as low as 0.50%. This fast neural model of the MOSFET can, therefore, be used to predict the dc characteristics of the device with physics-based simulation accuracies.

V. Circuit Optimization Using Neural-Network Models

ANN models for RF/microwave components can be used in circuit design and optimization. To achieve this, the neural models are first incorporated into circuit simulators. For designers who run the circuit simulator, the neural models can be used in a similar way as other models available in the simulator's library. An ANN model can be connected with other ANN models or with any other models in the simulator to form a high-level circuit. In this section, circuit optimization examples utilizing fast and accurate neural models are presented.

A. CPW Folded Double-Stub Filter

In this example, a CPW folded double-stub filter shown in Fig. 7 was designed. For this design, the substrate parameters ($\varepsilon_r = 12.9$, $H_{\rm sub} = 625$ μm, $\tan\delta = 0.0005$) and the CPW parameters ($W = 70$ μm, $G = 60$ μm) were fixed, yielding $Z_o \approx 50$ Ω. Fast neural-network models of CPW components, namely, the CPW transmission line, 90° compensated bends, short-circuit stubs, and symmetric T-junctions (the one described in Section IV) were trained using accurate training data from full-wave EM simulations. Design and optimization of the filter were accomplished using the *HP MDS*[5] network simulator and the EM-ANN models of various components [10].

The filter was designed for a center frequency of 26 GHz. Ideally, the length of each short-circuit stub and the section of line between the stubs should have a length of $\lambda/4$ at 26 GHz. However, due to the presence of discontinuities, these lengths need to be adjusted. Parameters to be optimized are $l_{\rm stub}$ and $l_{\rm mid}$. Initial values for these lengths were determined, and the corresponding structure showed a less than ideal response when simulated. The circuit was then optimized, using gradient-descent, to provide the desired circuit response. The effect of optimization was a reduction in the two line lengths. A comparison of circuit responses using the initial EM-ANN design, optimized EM-ANN design, and full-wave EM simulation of the optimized filter circuit are shown in Fig. 9. A good agreement was obtained between the EM-ANN and full-wave EM simulations of the optimized circuit over a frequency range of 1–50 GHz.

B. Three-Stage Monolithic-Microwave Integrated-Circuit (MMIC) Amplifier

In this example, MESFET neural models were used for yield optimization of a three-stage X-band MMIC amplifier shown in Fig. 10. Component ANN models were incorporated into circuit simulators Agilent *ADS*[6] and *OSA90*. Numerical results in this example were from the *OSA90* implementation [12]. The specifications for the amplifier are as follows.

- Passband (8–12 GHz): 12.4 dB $\leq$ gain $\leq$ 15.6 dB, Input voltage standing-wave ratio (VSWR) $\leq$ 2.8.
- Stopband (less than 6 GHz and greater than 15 GHz): gain $\leq$ 2 dB.

The ANN models of MESFETs (developed in Section IV) were used in this amplifier design. There are 14 design variables, i.e., metal-plate areas (S_{C_i}) of the metal–insulator–metal (MIM) capacitors (C_i, $i = 1, 2, 3, 4$) and number of turns (n_{L_j}) of the spiral inductors (L_j, $j =$ 1–10). A total of 37 statistical variables including gate length, gatewidth, channel thickness, and doping density of MESFET models, metal-plate area and thickness of capacitor models, and conductor width and spacing of spiral inductor models were considered.

[3]*OSA90, ver. 3.0,* Optimization Syst. Associates, Dundas, ON, Canada (now Agilent EEsof, Santa Rosa, CA).

[4]*MINIMOS, ver. 6.1,* Inst. Microelectron., Tech. Univ. Vienna, Vienna, Austria.

[5]*MDS,* Agilent Technol., Santa Rosa, CA.

[6]*ADS,* Agilent Technol., Santa Rosa, CA.

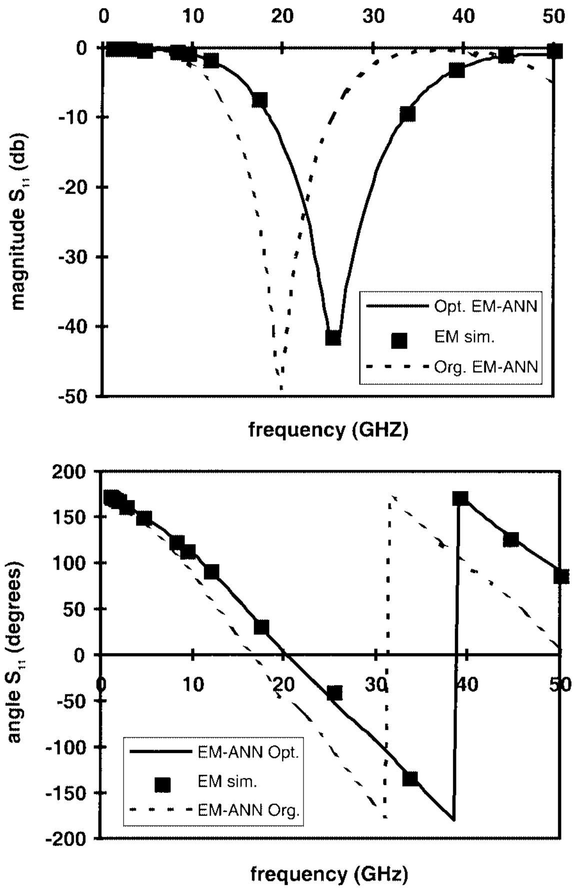


Fig. 9. Comparison of the CPW folded double-stub filter responses before and after ANN-based optimization. A good agreement is achieved between ANN-based simulations and full-wave EM simulations of the optimized circuit.

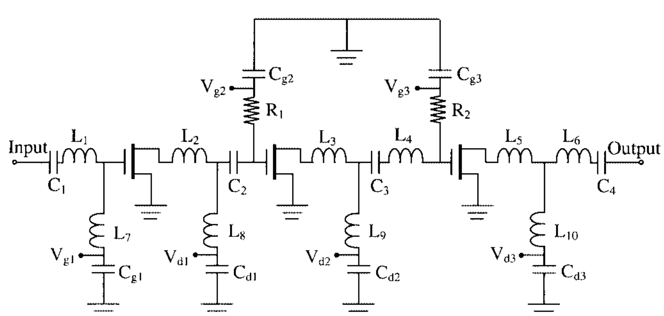


Fig. 10. A three-stage MMIC amplifier in which the three MESFETs are represented by neural-network models.

Yield optimization using an l_1-centering algorithm [28] was performed with a minimax nominal design solution as the initial point. The initial yield (before optimization) of the amplifier using the minimax nominal design was 26% with fast ANN-based simulations and 32% with relatively slow simulations using the Khatibzadeh and Trew MESFET models. After yield optimization using neural-network models, the amplifier yield improved from 32% to 58%, as verified by the Monte Carlo analysis using the original MESFET models. The Monte Carlo responses before and after yield optimization are shown in Fig. 11. The use of neural-network models instead of the Khatibzadeh and Trew models reduced the computation time for nonlinear statistical analysis and yield optimization from days to hours [12].

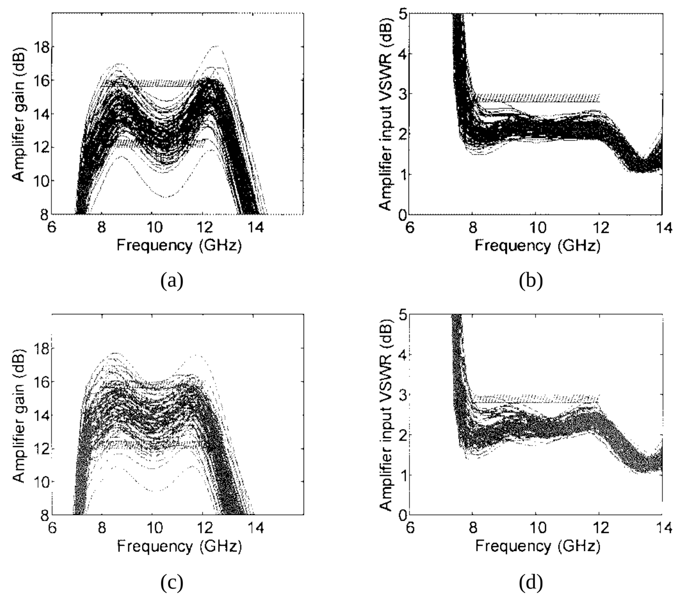


Fig. 11. Monte Carlo responses of the three-stage MMIC amplifier. (a) and (b) Before yield optimization. (c) and (d) After yield optimization. Yield optimization was carried out using neural-network models of MESFETs.

Considering that the Khatibzadeh and Trew models used in this example for illustration purpose are semianalytical in nature, the CPU speed-up offered by neural-based design relative to circuit design using physics-based semiconductor equations could be even more significant.

VI. Conclusions

Neural networks have recently gained attention as a fast, accurate, and flexible tool to RF/microwave modeling, simulation, and design. As this emerging technology expands from university research into practical applications, there is a need to address the basic conceptual issues in ANN-based CAD. Through this tutorial, we have tried to build a technical bridge between microwave design concepts and neural-network fundamentals. Principal ideas in neural-network-based techniques have been explained to design-oriented readers in a simple manner. Neural-network model development from beginning to end has been described with all the important steps involved. To demonstrate the application issues, a set of selected component modeling and circuit optimization examples have been presented. The ANN techniques are also explained through a multimedia presentation including narrative audio clips (Appendix I) in the electronic version of this paper on the CD-ROM accompanying this issue. For those readers interested in benefiting from neural networks right away, we have provided a hyperlink to *NeuroModeler* demonstration software (Appendix II).

Appendix I
Multimedia Slide Presentation

A multimedia Microsoft PowerPoint slide presentation including narrative audio clips is made available to the readers in the form of an Appendix. The presentation consisting of 55 slides provides systematic highlights of the microwave-ANN methodology and its practical applications. Some of the advanced concepts are simplified using slide-by-slide illustrations and animated transitions. The audio clips further help to make self-learning of this emerging area easier.

APPENDIX II
HYPERLINK TO *NeuroModeler* SOFTWARE

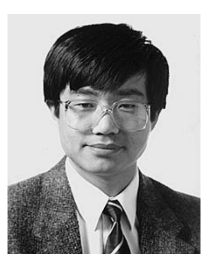
A hyperlink to the demonstration version of *NeuroModeler* software is provided. The software can be used to practice various interesting concepts in the tutorial including neural-network structure creation, neural-network training, neural model testing, etc. The main purpose is to enable the readers to better understand the neural-network-based design techniques and to get quick hands-on experience.

ACKNOWLEDGMENT

The authors thank L. Ton and M. Deo, both of the Department of Electronics, Carleton University, Ottawa, ON, Canada, for their help in preparing the multimedia Microsoft PowerPoint slide presentation and this paper's manuscript, respectively.

REFERENCES

[1] Q. J. Zhang and K. C. Gupta, *Neural Networks for RF and Microwave Design*. Norwood, MA: Artech House, 2000.
[2] K. C. Gupta, "Emerging trends in millimeter-wave CAD," *IEEE Trans. Microwave Theory Tech.*, vol. 46, pp. 747–755, June 1998.
[3] V. K. Devabhaktuni, M. Yagoub, Y. Fang, J. Xu, and Q. J. Zhang, "Neural networks for microwave modeling: Model development issues and nonlinear modeling techniques," *Int. J. RF Microwave Computer-Aided Eng.*, vol. 11, pp. 4–21, 2001.
[4] V. K. Devabhaktuni, M. Yagoub, and Q. J. Zhang, "A robust algorithm for automatic development of neural-network models for microwave applications," *IEEE Trans. Microwave Theory Tech.*, vol. 49, pp. 2282–2291, Dec. 2001.
[5] V. K. Devabhaktuni, C. Xi, F. Wang, and Q. J. Zhang, "Robust training of microwave neural models," *Int. J. RF Microwave Computer-Aided Eng.*, vol. 12, pp. 109–124, 2002.
[6] F. Wang and Q. J. Zhang, "Knowledge-based neural models for microwave design," *IEEE Trans. Microwave Theory Tech.*, vol. 45, pp. 2333–2343, Dec. 1997.
[7] F. Wang, V. K. Devabhaktuni, and Q. J. Zhang, "A hierarchical neural network approach to the development of a library of neural models for microwave design," *IEEE Trans. Microwave Theory Tech.*, vol. 46, pp. 2391–2403, Dec. 1998.
[8] P. M. Watson and K. C. Gupta, "EM-ANN models for microstrip vias and interconnects in dataset circuits," *IEEE Trans. Microwave Theory Tech.*, vol. 44, pp. 2495–2503, Dec. 1996.
[9] J. W. Bandler, M. A. Ismail, J. E. Rayas-Sanchez, and Q. J. Zhang, "Neuromodeling of microwave circuits exploiting space-mapping technology," *IEEE Trans. Microwave Theory Tech.*, vol. 47, pp. 2417–2427, Dec. 1999.
[10] P. M. Watson and K. C. Gupta, "Design and optimization of CPW circuits using EM-ANN models for CPW components," *IEEE Trans. Microwave Theory Tech.*, vol. 45, pp. 2515–2523, Dec. 1997.
[11] G. L. Creech, B. J. Paul, C. D. Lesniak, T. J. Jenkins, and M. C. Calcatera, "Artificial neural networks for fast and accurate EM-CAD of microwave circuits," *IEEE Trans. Microwave Theory Tech.*, vol. 45, pp. 794–802, May 1997.
[12] A. H. Zaabab, Q. J. Zhang, and M. S. Nakhla, "A neural network modeling approach to circuit optimization and statistical design," *IEEE Trans. Microwave Theory Tech.*, vol. 43, pp. 1349–1358, June 1995.
[13] Y. Fang, M. Yagoub, F. Wang, and Q. J. Zhang, "A new macromodeling approach for nonlinear microwave circuits based on recurrent neural networks," *IEEE Trans. Microwave Theory Tech.*, vol. 48, pp. 2335–2344, Dec. 2000.
[14] J. Xu, M. Yagoub, R. Ding, and Q. J. Zhang, "Neural-based dynamic modeling of nonlinear microwave circuits," *IEEE Trans. Microwave Theory Tech.*, vol. 50, pp. 2769–2780, Dec. 2002.
[15] M. Vai and S. Prasad, "Microwave circuit analysis and design by a massively distributed computing network," *IEEE Trans. Microwave Theory Tech.*, vol. 43, pp. 1087–1094, May 1995.
[16] M. Vai, S. Wu, B. Li, and S. Prasad, "Reverse modeling of microwave circuits with bidirectional neural network models," *IEEE Trans. Microwave Theory Tech.*, vol. 46, pp. 1492–1494, Oct. 1998.
[17] J. A. Jargon, K. C. Gupta, and D. C. DeGroot, "Applications of artificial neural networks to RF and microwave measurements," *Int. J. RF Microwave Computer-Aided Eng.*, vol. 12, pp. 3–24, 2002.
[18] P. M. Watson, C. Cho, and K. C. Gupta, "Electromagnetic-artificial neural network model for synthesis of physical dimensions for multilayer asymmetric coupled transmission structures," *Int. J. RF Microwave Computer-Aided Eng.*, vol. 9, pp. 175–186, 1999.
[19] F. Wang, V. K. Devabhaktuni, C. Xi, and Q. J. Zhang, "Neural network structures and training algorithms for microwave applications," *Int. J. RF Microwave Computer-Aided Eng.*, vol. 9, pp. 216–240, 1999.
[20] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks*, vol. 2, pp. 359–366, 1989.
[21] T. Y. Kwok and D. Y. Yeung, "Constructive algorithms for structure learning in feedforward neural networks for regression problems," *IEEE Trans. Neural Networks*, vol. 8, pp. 630–645, May 1997.
[22] J. de Villiers and E. Barnard, "Backpropagation neural nets with one and two hidden layers," *IEEE Trans. Neural Networks*, vol. 4, pp. 136–141, Jan. 1992.
[23] G. Thimm and E. Fiesler, "High-order and multilayer perceptron initialization," *IEEE Trans. Neural Networks*, vol. 8, pp. 349–359, Mar. 1997.
[24] A. Veluswami, M. S. Nakhla, and Q. J. Zhang, "The application of neural networks to EM-based simulation and optimization of interconnects in high-speed VLSI circuits," *IEEE Trans. Microwave Theory Tech.*, vol. 45, pp. 712–723, May 1997.
[25] R. Griffith and M. S. Nakhla, "Time-domain analysis of lossy coupled transmission lines," *IEEE Trans. Microwave Theory Tech.*, vol. 38, pp. 1480–1487, Oct. 1990.
[26] M. A. Khatibzadeh and R. J. Trew, "A large-signal, analytical model for the GaAs MESFET," *IEEE Trans. Microwave Theory Tech.*, vol. 36, pp. 231–239, Feb. 1988.
[27] V. K. Devabhaktuni, B. Chattaraj, M. Yagoub, and Q. J. Zhang, "Advanced microwave modeling framework exploiting automatic model generation, knowledge neural networks, and space mapping," in *Proc. IEEE MTT-S Int. Microwave Symp.*, Seattle, WA, 2002, pp. 1097–1100.
[28] J. W. Bandler and S. H. Chen, "Circuit optimization: The state of the art," *IEEE Trans. Microwave Theory Tech.*, vol. 36, pp. 424–443, MONTH 1988.



Qi-Jun Zhang (S'84–M'87–SM'95) received the B.Eng. degree from East China Engineering Institute, Nanjing, China, in 1982, and the Ph.D. degree in electrical engineering from McMaster University, Hamilton, ON, Canada, in 1987.

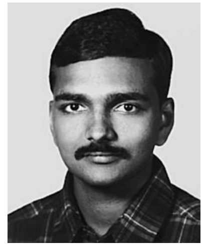
He was with the System Engineering Institute, Tianjin University, Tianjin, China, in 1982 and 1983. During 1988–1990, he was with Optimization Systems Associates Inc. (OSA), Dundas, ON, Canada, developing advanced microwave optimization software. In 1990, he joined the Department of Electronics, Carleton University, Ottawa, ON, Canada, where he is presently a Professor. His research interests are neural network and optimization methods for high-speed/high-frequency circuit design, and has authored more than 150 papers on these topics. He is a coauthor of *Neural Networks for RF and Microwave Design* (Boston, MA: Artech House, 2000), a Co-Editor of *Modeling and Simulation of High-Speed VLSI Interconnects* (Boston, MA: Kluwer, 1994), and a contributor to *Analog Methods for Computer-Aided Analysis and Diagnosis* (New York: Marcel Dekker, 1988). He was a Guest Co-Editor for a Special Issue on High-Speed VLSI Interconnects of the *International Journal of Analog Integrated Circuits and Signal Processing* and twice a Guest Editor for the Special Issues on Applications of ANN to RF and Microwave Design for the *International Journal of Radio Frequency and Microwave Computer-Aided Engineering*.

Dr. Zhang is a member of the Professional Engineers of Ontario, Canada.

Kuldip C. Gupta (M'62–SM'74–F'88) received the B.Sc. degree in physics, math, and chemistry from Punjab University, Punjab, India, in 1958, the B.E. and M.E. degrees in electrical communication engineering from the Indian Institute of Science, Bangalore, India, in 1961 and 1962, respectively, and the Ph.D. degree from the Birla Institute of Technology and Science, Pilani, India, in 1969.

Since 1983, he has been a Professor with the University of Colorado at Boulder. He is also currently the Associate Director for the National Science Foundation (NSF) Industry/University Cooperative Research (I/UCR) Center for Advanced Manufacturing and Packaging of Microwave, Optical and Digital Electronics (CAMPmode), University of Colorado at Boulder, and a Guest Researcher with the RF Technology Group, National Institute of Standards and Technology (NIST), Boulder, CO. From 1969 to 1984, he was with the Indian Institute of Technology (IITK), Kanpur, India, where he was a Professor of electrical engineering. From 1971 to 1979, he was the Coordinator for the Phased Array Radar Group, Advanced Center for Electronics Systems, Indian Institute of Technology. While on leave from the IITK, he was a Visiting Professor with the University of Waterloo, Waterloo, ON, Canada, the Ecole Polytechnique Federale de Lausanne, Lausanne, Switzerland, the Technical University of Denmark, Lyngby, Denmark, the Eidgenossische Technische Hochschule, Zurich, Switzerland, and the University of Kansas, Lawrence. From 1993 to 1994, while on sabbatical from the University of Colorado at Boulder, he was a Visiting Professor with the Indian Institute of Science and a consultant with the Indian Telephone Industries. His current research interests are the areas of CAD techniques (including ANN applications) for microwave and millimeter-wave ICs, nonlinear characterization and modeling, RF microelectromechanical systems (MEMS), and reconfigurable antennas. He has authored or coauthored *Microwave Integrated Circuits* (New York: Wiley, 1974; New York: Halsted Press (of Wiley), 1974), *Microstrip Line and Slotlines* (Norwood, MA: Artech House, 1979; revised 2nd edition, 1996), *Microwaves* (New York: Wiley, 1979; New York: Halsted Press (of Wiley), 1980, Mexico City, Mexico: Editorial Limusa Mexico, 1983), *CAD of Microwave Circuits* (Norwood, MA: Artech House, 1981, Beijing, China: Chinese Scientific Press, 1986, Moscow, Russia: Radio I Syvaz, 1987), *Microstrip Antenna Design* (Norwood, MA: Artech House, 1988), *Analysis and Design of Planar Microwave Components* (Piscataway, NJ: IEEE Press, 1994), *Analysis and Design of Integrated Circuit-Antenna Modules* (New York: Wiley 1999), and *Neural Networks for RF and Microwave Design* (Norwood, MA: Artech House 2000). He has also contributed chapters to the *Handbook of Microstrip Antennas* (Stevenage, U.K.: Peregrinus, 1989), the *Handbook of Microwave and Optical Components, Volume 1* (New York: Wiley, 1989), *Microwave Solid State Circuit Design* (New York: Wiley, 1988; 2nd edition 2003), *Numerical Techniques for Microwave and Millimeter Wave Passive Structures* (New York: Wiley, 1989), and the *Encyclopedia of Electrical and Electronics Engineering* (New York: Wiley, 1999). He has also authored or coauthored over 230 research papers. He holds four patents in the microwave area. He is the Founding Editor of the *International Journal of RF and Microwave Computer-Aided Engineering*, which is published by Wiley since 1991. He is on the Editorial Board of *Microwave and Optical Technology Letters* (Wiley), and the *International Journal of Numerical Modeling* (Wiley). He is listed in *Who's Who in America*, *Who's Who in the World*, *Who's Who in Engineering*, and *Who's Who in American Education*.

Dr. Gupta is a Fellow of the Institution of Electronics and Telecommunication Engineers (India), a member of URSI (Commission D, U.S.), and a member of the Electromagnetics Academy (Massachusetts Institute of Technology (MIT), Cambridge). He is a member of the Administrative Committee (AdCom) for the IEEE Microwave Theory and Techniques Society (IEEE MTT-S), chair of the IEEE MTT-S Standing Committee on Education, past co-chair of the IEEE MTT-S Technical Committee on Computer-Aided Design (MTT-1), a member of the IEEE Technical Committee on Microwave Field Theory (MTT-15), an earlier member of the IEEE-EAB Committee on Continuing Education and the IEEE-EAB Societies Education Committee. He is an associate editor for *IEEE Microwave Magazine* and is on the Editorial Board of the IEEE TRANSACTIONS ON MICROWAVE THEORY AND TECHNIQUES. He was a recipient of the IEEE Third Millennium Medal and the IEEE MTT-S Distinguished Educator Award.



Vijay K. Devabhaktuni (S'97) received the B.Eng. degree in electrical and electronics engineering and the M.Sc. degree in physics from the Birla Institute of Technology and Science, Pilani, Rajasthan, India, in 1996, and is currently working toward the Ph.D. degree in electronics at Carleton University, Ottawa, ON, Canada.

He is currently a Sessional Lecturer with the Department of Electronics, Carleton University. His research interests include ANNs, computer-aided-design methodologies for VLSI circuits, and RF and microwave modeling techniques.

Mr. Devabhaktuni was the recipient of a 1999 Best Student Research Exhibit Award presented by Nortel Networks. He was a two-time recipient of the Ontario Graduate Scholarship for the 1999–2000 and 2000–2001 academic years presented by the Ministry of Education and Training, ON, Canada. He was the recipient of the 2001 John Ruptash Memorial Fellowship, which is awarded annually to an outstanding graduate student of the Faculty of Engineering, Carleton University. He was also the recipient of the 2001 Teaching Excellence Award in Engineering presented by the Carleton University Student's Association.