Knowledge-Based Neural Models for Microwave Design

Fang Wang, *Student Member, IEEE*, and Qi-jun Zhang, *Senior Member, IEEE*

***Abstract*— Neural networks have recently been introduced to the microwave area as a fast and flexible vehicle to microwave modeling, simulation and optimization. In this paper, a novel neural network structure, namely, knowledge-based neural network (KBNN), is proposed where microwave empirical or semi-analytical information is incorporated into the internal structure of neural networks. The microwave knowledge complements the capability of learning and generalization of neural networks by providing additional information which may not be adequately represented in a limited set of training data. Such knowledge becomes even more valuable when the neural model is used to extrapolate beyond training data region. A new training scheme employing gradient based l_2 optimization technique is developed to train the KBNN model. The proposed technique can be used to model passive and active microwave components with improved accuracy, reduced cost of model development and less need of training data over conventional neural models for microwave design.**

***Index Terms*— Computer-aided design, modeling, neural network, optimization, simulation.**

I. Introduction

THE DRIVE for manufacturability-oriented design and reduced time-to-market in the microwave industry requires design tools that are accurate and fast. Statistical analysis and optimization with detailed physics/electromagnetic (EM) models of active/passive components can be an important step toward a design for first-pass success, but it is also computationally prohibitive using conventional computer-aided design (CAD) techniques. In recent years, a new CAD approach based on neural network models has been introduced for microwave design. It has been applied to the efficient modeling of microwave components, e.g., microstrip interconnects [1]–[3], vias [2], spiral inductors [4], and FET devices [5], [6]; and to the analysis and design of microwave circuits, e.g., microstrip circuit design [7], automatic microwave impedance matching [8] and Smith chart oriented microwave circuit analysis and design [9]. The neural network approach has also been applied to circuit optimization and statistical design, e.g., signal integrity analysis and optimization of very large scale integrated circuit (VLSI) interconnects [1], [3], microwave circuit optimization and statistical design with neural network models at either device or circuit levels [6], [10]. These pioneering works helped to establish the framework and the benefits of the neural modeling approach for microwave applications. Neural models can be much faster than original detailed EM/physics models [2], [10], more accurate than polynomial and empirical models [11], allow more dimensions than table lookup models [12] and are easier to develop when a new device/technology is introduced [13].

This paper addresses some of the growing demands in the continuing application of neural networks in microwave design, i.e., reductions of model development cost and improvement of model reliability. The most widely used form of neural networks is the multilayer perceptron (MLP) network which has been adopted in [1]–[7], [10]. MLP has a well-established error-backpropagation training algorithm, is simple and has been used in many applications such as signal processing [14], control [15], speech recognition [16], as well as in microwave design [1]–[7], [10]. Since the MLP belongs to the type of black-box model structurally embedding no problem dependent information, the entire information about the application comes from training data. Consequently, large amount of training data is usually needed to ensure model accuracy. In our microwave application, training data is obtained by either simulation of the original EM or device physics problem, or by measurement. Generating large amount of training data could be very expensive for microwave problems because simulation/measurement may have to be performed for many combinations of different values of geometrical/material/process parameters in the EM or device physics problems. Without sufficient training data, the neural models developed may not be very reliable. In addition, even with sufficient training data, the reliability of MLP when used for extrapolation purpose is not guaranteed and in many cases is very poor.

Several attractive approaches to improve neural network accuracy/generalization capability have been proposed in the past in the microwave and the neural network communities. Selection of suitable size of the neural models has been one way to improve model generalization capability, e.g., network pruning [17]. In [2], a hybrid EM–ANN model was proposed using microwave empirical formulas to approximate the original model, and using MLP to model the difference between the original simulation and empirical models. Another approach is to add prior knowledge into neural networks, e.g., [18]. Such knowledge provides additional information of the original problem which may not be adequately represented in the limited training data. One type of the existing approaches is to use symbolic knowledge in the form of rules to establish

Manuscript received March 31, 1997; revised July 21, 1997. This work was supported by the Natural Sciences and Engineering Research Council of Canada and by Micronet: A Canadian Network Centres of Excellence on Microelectronic Devices, Circuits and Systems.

The authors are with the Department of Electronics, Carleton University, Ottawa, ON K1S 5B6, Canada.

Publisher Item Identifier S 0018-9480(97)08332-4.

the structure and weights in a neural network [18], [19]. The weights, e.g., the certainty factor associated with rules [20] or both the topology and weights of the network [21] can be revised during training. Other approaches to build prior information into neural networks are, e.g., restricting the network architecture through the use of local connections and constraining the choice of weights by the use of weight sharing [14]. Unfortunately, the existing approaches to incorporate knowledge are largely using symbolic information and are often oriented to pattern recognition area. In microwave modeling areas, however, the most important problem knowledge is more functional than symbolic/structural [22], [23], making the existing knowledge network methods unsuitable for microwave applications.

In this paper, we propose a new microwave-oriented knowledge-based neural network (KBNN) [24] in which microwave knowledge in the form of empirical functions or analytical approximations is embedded into internal neural network structures. Switching boundary and region neurons are introduced in the model structure to reflect microwave cases where different equations or formulas with different parameters can be interchangeably used in different regions of the input parameter space. The proposed structure does not follow the rigorous layer-by-layer structure in MLP, and a new training algorithm is developed since conventional backpropagation is not applicable. The proposed technique enhances neural model accuracy especially for unseen data and reduces the need of large set of training data.

This paper is organized as follows. Section II starts with the problem statement of microwave design using neural networks and then presents the proposed structure of knowledge-based neural networks (KBNN's) including a description of the functionality of individual neurons in the network. A new training approach for the proposed neural network structure is described in Section III. Section IV presents several examples to demonstrate the features and advantages of the proposed neural model, including circuit waveform modeling, transmission line and MESFET modeling examples.

II. Proposed Knowledge-Based Neural Network (KBNN) Structure

A. Neural-Based Microwave Modeling: Problem Statement

Let $\mathbf{x}$ be a N_x-vector containing parameters of a given device or a circuit, e.g., gate length and gate width of a FET or geometrical and physical parameters of transmission lines. Let $\mathbf{y}$ be a N_y-vector including the responses of the device or the circuit under consideration, e.g., drain current of a FET. The relationship between $\mathbf{x}$ and $\mathbf{y}$ is multidimensional and nonlinear. Such a relationship can be modeled by a neural network which typically consists of a collection of interconnected neurons, e.g., a conventional MLP [10]. For example, a three-layer perceptron shown in Fig. 1, consists of an input layer and an output layer, corresponding to model input and output variables $\mathbf{x}$ and $\mathbf{y}$, respectively, as well as a hidden layer.

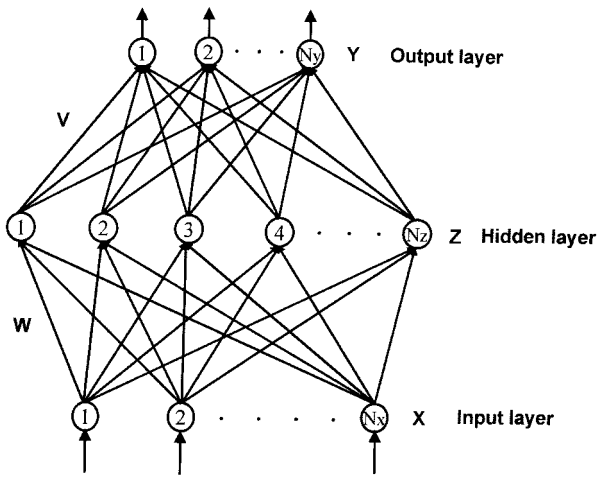


Fig. 1. The multilayer perceptrons structure. Shown here is a popularly used three-layer MLP.

Given input $\mathbf{x}$, the output can be computed by

$$y_j = \sum_{k=1}^{N_z} z_k v_{jk} + \eta, \quad j = 1, \cdots, N_y \tag{1}$$

where z_k are values of hidden neurons computed as

$$\begin{aligned} z_k &= g(\gamma_k) \\ \gamma_k &= \left(\sum_{i=1}^{N_x} x_i w_{ki} \right) + \theta_k \end{aligned} \tag{2}$$

where $g(\cdot)$ is an activation function. The overall nonlinear relationship between $\mathbf{x}$ and $\mathbf{y}$ is realized by various activation patterns of the neurons whose activation functions are typically a smooth switch function, e.g., the sigmoid function

$$g(\gamma) = \frac{1}{1 + e^{-\gamma}}. \tag{3}$$

In the model development stage, samples of $(\mathbf{x}, \mathbf{y})$ data, called training data, are generated from original EM simulation or measurement. The neural model is then trained to learn the input–output $(\mathbf{x}$–$\mathbf{y})$ relationship from the training data. Specifically training is to determine neural model parameters, i.e., neural network internal weights w_{ki} and v_{jk}, such that the neural model predicted output best matches that of training data. In the model testing stage, a new set of input–output samples, called testing data, is used to test the accuracy of the neural model. The ability of neural models to give accurate $\mathbf{y}$ when presented with input parameter values $\mathbf{x}$ never seen during training is called the generalization ability. A trained and tested neural model can then be used online during microwave design stage providing fast model evaluation replacing original slow EM/device simulators. The benefit of the neural model approach is especially significant when the model is highly repetitively used such as in optimization, Monte Carlo analysis and yield maximization.

The conventional neural networks, e.g., MLP, are blind black box models. The standard switching activation functions such as sigmoid or hyperbolic tangent functions used in MLP are far different from the various engineering models.

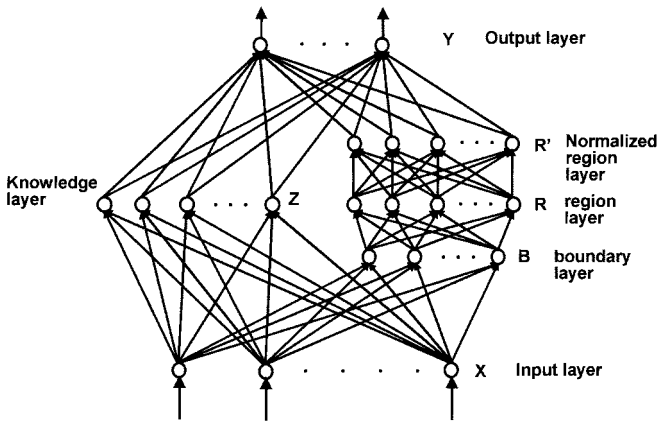


Fig. 2. The proposed KBNN structure.

The generalization ability of the MLP model comes only from the information encoded in the training data. Large amount of training data is usually required to ensure model accuracy which means increased cost of model development due to large amount of simulation/measurement of original EM/device physics problems.

In the next subsection, we will introduce a new neural model structure to incorporate the available engineering knowledge about the circuit/components in the form of empirical/semi-analytical formulas into neural models.

B. Proposed KBNN Structure

The proposed KBNN structure is a nonfully connected structure shown in Fig. 2. There are six layers in the structure, namely input layer $\mathbf{X}$, knowledge layer $\mathbf{Z}$, boundary layer $\mathbf{B}$, region layer $\mathbf{R}$, normalized region layer $\mathbf{R}'$ and output layer $\mathbf{Y}$. The input layer $\mathbf{X}$ accepts parameters $\mathbf{x}$ from outside the model. The knowledge layer $\mathbf{Z}$ is the place where microwave knowledge resides in the form of single or multidimensional functions $\Psi(\cdot)$. For knowledge neuron i in the $\mathbf{Z}$ layer:

$$z_i = \Psi_i(\mathbf{x}, \mathbf{w}_i), \quad i = 1, 2, \cdots, N_z \tag{4}$$

where $\mathbf{x}$ is a vector including neural network inputs x_i, $i = 1, 2, \cdots, N_x$ and $\mathbf{w}_i$ is a vector of parameters in the knowledge formula. The knowledge function $\Psi_i(\mathbf{x}, \mathbf{w}_i)$ is usually in the form of empirical or semi-analytical functions, for example, the drain current of a FET as a function of its gate length, gate width, channel thickness, doping density, gate voltage and drain voltage [23]. The boundary layer $\mathbf{B}$ can incorporate knowledge in the form of problem dependent boundary functions $B(\cdot)$ or in the absence of boundary knowledge just as linear boundaries. Neuron i in this layer is calculated by

$$b_i = B_i(\mathbf{x}, \mathbf{v}_i), \quad i = 1, 2, \cdots, N_b \tag{5}$$

where $\mathbf{v}_i$ is a vector of parameters in B_i defining an open or closed boundary in the input space $\mathbf{x}$. Let $\sigma(\cdot)$ be a sigmoid function. The region layer $(\mathbf{R})$ contains neurons to construct regions from boundary neurons,

$$r_i = \prod_{j=1}^{N_b} \sigma(\alpha_{ij} b_j + \theta_{ij}), \quad i = 1, 2, \cdots, N_r \tag{6}$$

where α_{ij} and θ_{ij} are the scaling and bias parameters, respectively. The normalized region layer $\mathbf{R}'$ contains rational function based neurons [25] to normalize the outputs of region layer,

$$r'_i = \frac{r_i}{\sum_{j=1}^{N_r} r_j}, \quad i = 1, 2, \cdots, N_{r'},\ N_{r'} = N_r. \tag{7}$$

The output layer $\mathbf{Y}$ contains second-order neurons [26] combining knowledge neurons and normalized region neurons,

$$y_j = \sum_{i=1}^{N_z} \beta_{ji} z_i \left(\sum_{k=1}^{N_{r'}} \rho_{jik} r'_k \right) + \beta_{j0}, \quad j = 1, 2, \cdots, N_y \tag{8}$$

where β_{ji} reflects the contribution of the ith knowledge neuron to output neuron y_j and β_{j0} is the bias parameter. ρ_{jik} is 1 indicating that region r'_k is the effective region of the ith knowledge neuron contributing to the jth output. A total of $N_{r'}$ regions are shared by all the output neurons. As a special case, if we assume that each normalized region neuron selects a unique knowledge neuron for each output j, the function for output neurons can be simplified as

$$y_j = \sum_{i=1}^{N_z} \beta_{ji} z_i r'_i + \beta_{j0}, \quad j = 1, 2, \cdots, N_y \tag{9}$$

The prior knowledge encoded in $\Psi(\cdot)$ and/or $B(\cdot)$ needs not to be very accurate and complete. Several forms of functions $\Psi(\cdot)$ and/or $B(\cdot)$ can coexist in the network. The constant coefficients in the original empirical functions can be replaced by trainable parameters and more bias/scale parameters can be added to provide extra variability among different neurons. If some input parameters are not present in the original empirical models, they can be added to the knowledge function $\Psi(\cdot)$ in the weighted sum form. The proposed structure was inspired from the fact that practical empirical functions are usually valid only in a certain region of the parameter space. To build a neural model for the entire space, several empirical formulas and the mechanism to switch among them are needed. The switching mechanism expands the feature of sigmoidal radial basis function [27] into high-dimensional space and with more generalized activation functions. This model retains the essence of neural networks in that the exact location of each switching boundary, and the scale and position of each knowledge function are initialized randomly and then determined eventually during training.

III. Training Approach of KBNN

Let $\mathbf{y}$ represent the neural model output. Let $\bar{\mathbf{y}}$ represent the corresponding outputs from the original problem, e.g., original EM simulation or measurement. Neural network learns from a set of training data $(\mathbf{x_p}, \bar{\mathbf{y}}_\mathbf{p})$, $p = 1, \cdots, P$, where P is the total number of data samples. The trainable parameters of the proposed KBNN, denoted as Φ, includes $\mathbf{w}_i$, $i = 1, \cdots, N_z$, $\mathbf{v}_i$, $i = 1, \cdots, N_b$, α_{ij} and θ_{ij}, $i = 1, \cdots, N_r$, $j = 1, \cdots, N_b$, β_{ji}, β_{j0} and ρ_{jik}, $k = 1, \cdots, N_{r'}$, $i = 1, \cdots, N_z$, $j = 1, \cdots, N_y$. The purpose of training is to determine the weight parameters Φ inside the KBNN model such that the error

between the desired outputs $\bar{\mathbf{y}}$ and the actual outputs $\mathbf{y}$ from KBNN is minimized,

$$\min_{\Phi} \frac{1}{2} \sum_{p=1}^{P} \sum_{j=1}^{N_y} (y_{pj} - \bar{y}_{pj})^2. \tag{10}$$

A gradient based l_2 optimization technique is employed in the training of KBNN, which requires the derivative of error from individual training samples with respect to each weight in KBNN, i.e., a Jacobian matrix $\mathbf{J}$. Since our network does not follow a regularly layered MLP structure, microwave empirical functions instead of standard activation functions are used in neurons, conventional backpropagation algorithm is not applicable. A new scheme extending the error backpropagating idea is derived to obtain this Jacobian matrix.

The error backpropagation path starts from the output layer $\mathbf{Y}$. Let the error of neural network prediction for a sample, e.g., the pth sample, be

$$\epsilon_p = \frac{1}{2} \sum_{j=1}^{N_y} (y_{pj} - \bar{y}_{pj})^2 \tag{11}$$

where y_{pj} is the actual jth neural network output and $\bar{y}_{pj}$ is the jth output of the pth training sample. Jacobian matrix $\mathbf{J}$ requires the derivative of error ϵ_p with respect to each weight in KBNN for each p. For simplicity, the subscript p is dropped in the following description. Let the derivative of ϵ with respect to the output of individual neurons be denoted as g. For output layer ($\mathbf{Y}$ layer), g_{y_j} is defined as $g_{y_j} = \partial\epsilon/\partial y_j$, which can be easily calculated from error function (11), as

$$\frac{\partial \epsilon}{\partial y_j} = y_j - \bar{y}_j, \quad j = 1, 2, \cdots, N_y. \tag{12}$$

Then the derivatives of error ϵ with respect to weights β's and ρ's inside the output neuron are obtained as

$$\frac{\partial \epsilon}{\partial \beta_{ji}} = (y_j - \bar{y}_j) z_i \left(\sum_{k=1}^{N_{r'}} \rho_{jik} r'_k \right), \quad i = 1, 2, \cdots, N_z, \quad j = 1, 2, \cdots, N_y \tag{13}$$

$$\frac{\partial \epsilon}{\partial \beta_{j0}} = (y_j - \bar{y}_j), \quad j = 1, 2, \cdots, N_y \tag{14}$$

$$\frac{\partial \epsilon}{\partial \rho_{jik}} = (y_j - \bar{y}_j) \beta_{ji} z_i r'_k, \quad k = 1, 2, \cdots, N_{r'}, \quad i = 1, 2, \cdots, N_z, \quad j = 1, 2, \cdots, N_y. \tag{15}$$

The proposed KBNN training scheme begins to differ from conventional backpropagation below the output layer $\mathbf{Y}$, where the error propagation is split into two paths, one through the knowledge layer $\mathbf{Z}$ down to the input layer $\mathbf{X}$, and the other through the normalized region layer $\mathbf{R}'$, the region layer $\mathbf{R}$ and the boundary layer $\mathbf{B}$ down to the input layer $\mathbf{X}$. In the first path, g_z can be obtained as,

$$g_{z_i} = \sum_{j=1}^{N_y} \frac{\partial \epsilon}{\partial y_j} \frac{\partial y_j}{\partial z_i} = \sum_{j=1}^{N_y} (y_j - \bar{y}_j) \beta_{ji} \left(\sum_{k=1}^{N_{r'}} \rho_{jik} r'_k \right), \quad i = 1, 2, \cdots, N_z. \tag{16}$$

Continuing the derivative chain rule, the derivatives of error ϵ with respect to the weights inside knowledge neurons (i.e., $\mathbf{w}_i$) are

$$\frac{\partial \epsilon}{\partial \mathbf{w}_i} = \frac{\partial \epsilon}{\partial z_i} \frac{\partial z_i}{\partial \mathbf{w}_i} = g_{z_i} \frac{\partial \Psi_i}{\partial \mathbf{w}_i}, \quad i = 1, 2, \cdots, N_z \tag{17}$$

where $\frac{\partial \Psi_i}{\partial \mathbf{w}_i}$ is obtained from problem-dependent microwave empirical functions. In the second path, $g_{r'}$ is first obtained,

$$g_{r'_k} = \sum_{j=1}^{N_y} \frac{\partial \epsilon}{\partial y_j} \frac{\partial y_j}{\partial r'_k} = \sum_{j=1}^{N_y} (y_j - \bar{y}_j) \sum_{i=1}^{N_z} \beta_{ji} z_i \rho_{jik}, \quad k = 1, 2, \cdots, N_{r'}. \tag{18}$$

The g's for the next two layers, i.e., $\mathbf{R}$ and $\mathbf{B}$ layers, are

$$g_{r_i} = \sum_{j=1}^{N_{r'}} \frac{\partial \epsilon}{\partial r'_j} \frac{\partial r'_j}{\partial r_i} = g_{r'_i} \frac{1}{\sum_{k=1}^{N_r} r_k} - \frac{1}{\left(\sum_{k=1}^{N_r} r_k \right)^2} \left(\sum_{j=1}^{N_{r'}} g_{r'_j} r_j \right), \quad i = 1, 2, \cdots, N_r \tag{19}$$

$$g_{b_i} = \sum_{j=1}^{N_r} \frac{\partial \epsilon}{\partial r_j} \frac{\partial r_j}{\partial b_i} = \sum_{j=1}^{N_r} g_{r_j} r_j (1 - \sigma(\alpha_{ji} b_i + \theta_{ji})) \alpha_{ji}, \quad i = 1, 2, \cdots, N_b. \tag{20}$$

The derivatives of error ϵ with respect to weights α's and θ's inside region neurons are

$$\frac{\partial \epsilon}{\partial \alpha_{ij}} = \frac{\partial \epsilon}{\partial r_i} \frac{\partial r_i}{\partial \alpha_{ij}} = g_{r_i} r_i (1 - \sigma(\alpha_{ij} b_j + \theta_{ij})) b_j, \quad i = 1, 2, \cdots, N_r, \quad j = 1, 2, \cdots, N_b \tag{21}$$

$$\frac{\partial \epsilon}{\partial \theta_{ij}} = \frac{\partial \epsilon}{\partial r_i} \frac{\partial r_i}{\partial \theta_{ij}} = g_{r_i} r_i (1 - \sigma(\alpha_{ij} b_j + \theta_{ij})), \quad i = 1, 2, \cdots, N_r, \quad j = 1, 2, \cdots, N_b. \tag{22}$$

The derivatives of error ϵ with respect to weights $\mathbf{v}_i$'s inside boundary neurons are,

$$\frac{\partial \epsilon}{\partial \mathbf{v}_i} = \frac{\partial \epsilon}{\partial b_i} \frac{\partial b_i}{\partial \mathbf{v}_i} = g_{b_i} \frac{\partial B_i}{\partial \mathbf{v}_i}, \quad i = 1, 2, \cdots, N_b. \tag{23}$$

The derivatives of error ϵ with respect to all the weights inside KBNN are thus calculated. From (13)–(15), (17), (21)–(23), Jacobian matrix $\mathbf{J}$ can be constructed. The next formula shows the the merge of two error back-propagation paths at the input layer $\mathbf{X}$

$$g_{x_i} = \frac{\partial \epsilon}{\partial x_i} = \sum_{j=1}^{N_z} \frac{\partial \epsilon}{\partial z_j} \frac{\partial z_j}{\partial x_i} + \sum_{j=1}^{N_b} \frac{\partial \epsilon}{\partial b_j} \frac{\partial b_j}{\partial x_i} = \sum_{j=1}^{N_z} g_{z_j} \frac{\partial \Psi_j}{\partial x_i} + \sum_{j=1}^{N_b} g_{b_j} \frac{\partial B_j}{\partial x_i}, \quad i = 1, 2, \cdots, N_x. \tag{24}$$

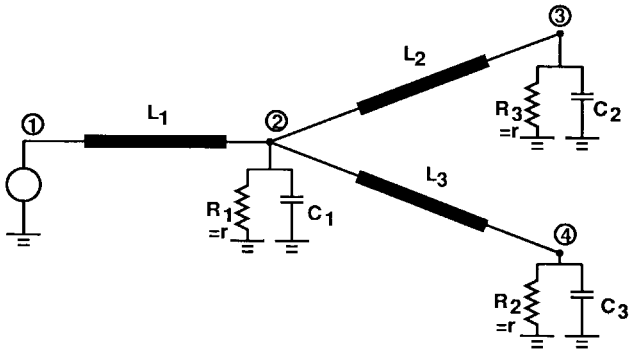


Fig. 3. The circuit for Example 1 with 3 transmission lines representing high-speed VLSI interconnects.

Jacobian matrix $\mathbf{J}$ have P columns with the pth column representing derivatives of the error ϵ_p from the pth sample with respect to all the weights in the network, showing as

$$\mathbf{J} = \left[\frac{\partial \epsilon_1}{\partial \Phi} \ \cdots \ \frac{\partial \epsilon_p}{\partial \Phi} \ \cdots \ \frac{\partial \epsilon_P}{\partial \Phi}\right]. \tag{25}$$

With the Jacobian matrix $\mathbf{J}$, the l_2 optimization approach [28] is employed in the training to provide better training efficiency over that of the conventional error backpropagation algorithm. The training is done with the entire set of training samples in one batch instead of conventional sample by sample backpropagation for MLP. The batch training approach can enjoy the use of powerful optimization techniques such as quasi-Newton and Luenberg–Marquart techniques [28] and is much more efficient especially if the training data need not to be massive, which is the case for KBNN.

IV. Examples

A. Example 1: Circuit Waveform Modeling

This simple example is mainly for illustration purpose showing the concept of incorporation of electrical knowledge into KBNN. A simple circuit with three transmission lines is shown in Fig. 3 representing signal integrity analysis of high-speed VLSI interconnects and terminations [13]. The excitation of the circuit is a step voltage source with 0.1 ns rise time at node 1. Neural networks (both KBNN and MLP) were used to model the waveform output at nodes 3 and 4. The inputs of neural models are resistor value r and time t, and the outputs are output voltage v_3 and v_4. As a circuit knowledge, the waveform can be estimated by a two-pole approximation with exponentially decayed sinusoids [29]. This two-pole approximation knowledge, plus an additional resistor variable r as part of the model input, was incorporated into KBNN by providing the knowledge function $\Psi(\cdot)$ at layer $\mathbf{Z}$ as

$$\begin{aligned} z_i &= \Psi_i(r, t, \mathbf{w}_i) \\ &= e^{-(w_{i1}r + w_{i2})t} \sin((w_{i3}r + w_{i4})t + w_{i5}r + w_{i6}), \\ &\qquad i = 1, 2, \cdots, N_z. \end{aligned} \tag{26}$$

With this two-pole approximation, the boundary in the parameter space is independent of time t. This is the knowledge embedded in layer $\mathbf{B}$ realized as a linear function of resistance only. The output layer is constructed following (9). The size of KBNN is represented by the number of neurons in $\mathbf{B}$ and $\mathbf{Z}$ layers, e.g., b1z2 representing a KBNN with one boundary neurons and two knowledge neurons. For the traditional MLP, the size of model is represented by the number of hidden neurons in the network, e.g., 7 representing a three-layer MLP with seven hidden neurons.

TABLE I
Model Accuracy Comparison Between Standard MLP and KBNN for Circuit Waveform Modeling Example. The Results Shown Are the Average of Three Different Trainings for Each Model

Neural net type	Model Size	No. of Weights	Average Test error	Largest Test error	Correlation Coefficient
Standard (MLP)	12	62	2.40%	18.00%	0.9449
	20	102	2.06%	13.52%	0.9641
	28	142	1.65%	9.13%	0.9784
	36	182	2.04%	12.98%	0.9664
Knowledge based (KBNN)	b1z4	36	1.00%	8.71%	0.9894

The training and testing data were obtained by simulation of the original circuit using HSPICE in the time interval from 0 to 7 ns. The training data was created with five resistor values of 25, 50, 75, 100, and 125 Ω and with only 10 time points per resistor value and is used to train the neural models (both KBNN and MLP). A KBNN of size b1z2 and four MLP's of sizes 12, 20, 28, and 36 were trained. Testing data includes 69 time points in the waveform on a different set of resistor values (r = 35, 55, 70, 90, and 115 Ω).

Table I shows the testing results of both models. For each model, the average accuracy from three trainings with different random starting points were used. The accuracy of the model is represented by the error and correlation coefficient between neural model output and testing data. A value of correlation coefficient closer to 1 indicates good accuracy of neural model. As seen in the table, the training data with 10 time points per resistor value were insufficient for MLP to model this set of waveforms. Fig. 4 shows the waveforms from original HSPICE simulation, KBNN and MLP for resistor value of 115 Ω. The waveforms indicate that the MLP model does not match well with original waveforms. With the same set of insufficient training data, KBNN shows very good accuracy. This illustrates that the prior knowledge provides additional information which is not adequately represented in the original training data. The incorporation of such knowledge into neural models is very helpful to produce a reliable model especially when fewer training data is available. Another interesting point is that the empirical knowledge alone, i.e., two-pole approximation with decay sinusoids, is not an adequate model by itself, since it cannot represent the waveform change with respect to resistor values. This example illustrates that through KBNN a simple empirical function can be used in a large parameter space provided that variations of the function with different sets of parameters are used in different regions of the space. The smooth switching between the regions is realized in the KBNN network by region neurons.

B. Example 2: Transmission Line Modeling

This example demonstrates the proposed KBNN in modeling cross sectional RLCG parameters of transmission lines

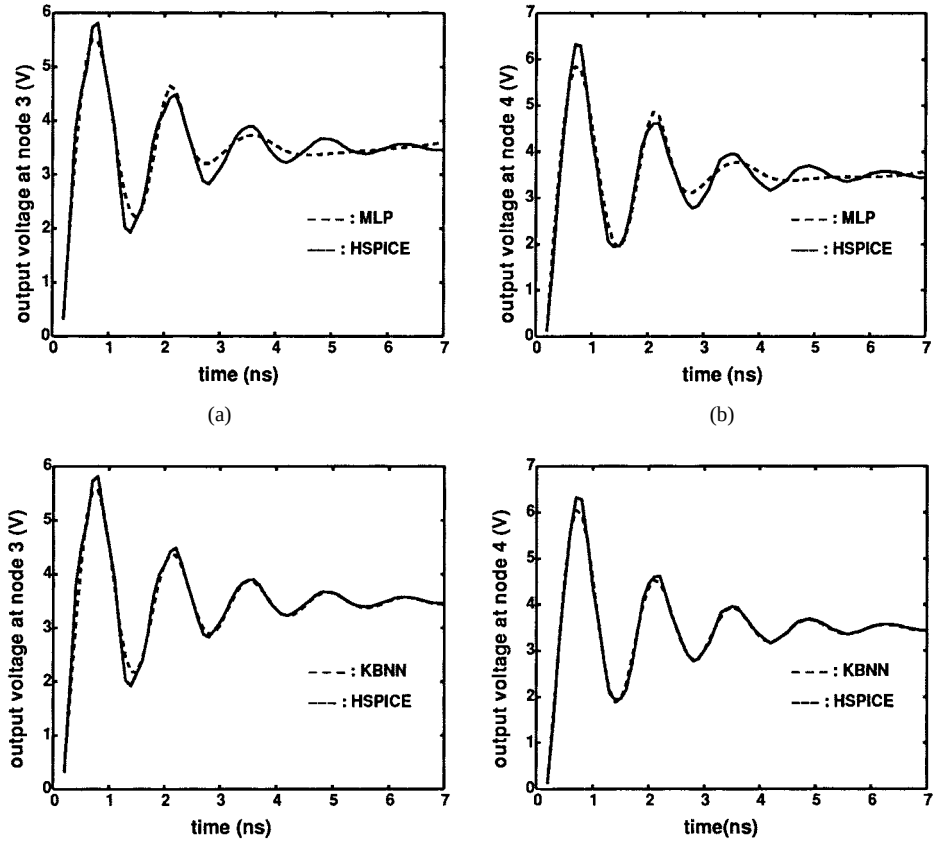


Fig. 4. Model accuracy comparison with original HSPICE simulation for circuit modeling example: (a) MLP and (c) KBNN at node 3; (b) MLP and (d) KBNN at node 4. All models were trained with data from only 10 points per waveform.

for analysis of high speed VLSI interconnects [13] and its comparison with traditional MLP. EM simulation of transmission lines is slow especially if it needs to be repeatedly evaluated. Neural networks learned from EM data have been found several hundred times faster than original EM simulation [3]. In this example, MLP and KBNN were used to model the cross-sectional per unit length mutual inductance, l_{12}, between two conductors of a coupled microstrip transmission line. The inputs of the neural models are width of conductor (x_1), thickness of conductor (x_2), separation between two conductors (x_3), height of substrate (x_4), relative dielectric constant (x_5) and frequency (x_6). There exist mutual inductance empirical formulas, e.g., [22],

$$l_{12} = \frac{\mu_r \mu_0}{4\pi} \ln\left[1 + \frac{(2x_4)^2}{(x_1 + x_3)^2}\right]. \tag{27}$$

This equation becomes the knowledge to be incorporated into the knowledge neurons following Section II as

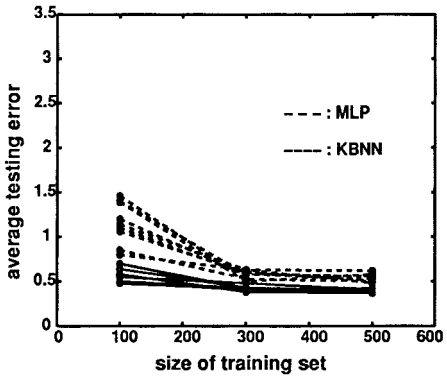
$$\begin{aligned} z_i &= \Psi_i(\mathbf{x}, \mathbf{w}_i) \\ &= \ln\left[1 + e^{w_{i1}} \frac{(x_4 - w_{i2})^2}{(x_1 + x_3 - w_{i3})^2}\right] \\ &\quad + w_{i4}x_2 + w_{i5}x_5 + w_{i6}x_6 + w_{i7}, \quad i = 1, 2, \cdots, N_z. \end{aligned} \tag{28}$$

Linear boundary neurons were used in the layer $\mathbf{B}$. Notice that this empirical formula is incorporated multiple times (N_z times), each with different values of $\mathbf{w}$, ($\mathbf{w}_i$, $i = 1, 2, \cdots, N_z$). KBNN provides a complete/integrated (**x-y**) relationship including those not available in the original empirical formula (e.g., y with respect to x_2, x_5, x_6).

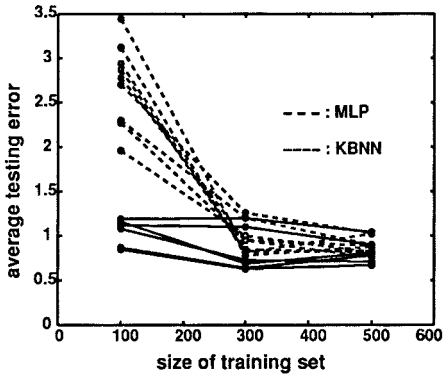
TABLE II
RANGES OF TRAINING DATA FOR NEURAL MODEL INPUT PARAMETERS FOR THE TRANSMISSION LINE MODELING EXAMPLE

Parameters	Notation	Range
Conductor Width	x_1	0.10 – 0.25 *mm*
Conductor Thickness	x_2	17 – 71 μm
Conductor Separations	x_3	0.10 – 0.76 *mm*
Substrate Height	x_4	0.10 – 0.31 *mm*
Relative Dielectric Constant	x_5	3.7 – 4.8
Frequency	x_6	0.5 – 2 *GHz*

Two KBNN's (of sizes b2z3 and b4z6) were built and compared with three MLP's (with number of hidden neurons being 7, 15, and 20). Five sets of data were generated by EM simulation [30]. The first three sets with 100, 300, and 500 samples were generated within the parameter range shown in Table II and were used for training purpose. The neural net training was done on SPARC station 5. The CPU time for MLP training by the conventional sample-by-sample error backpropagation approach ranged from 10 min (for small neural network with 100 training samples) to 45 min (for large neural network with 500 training samples). The CPU time for MLP or KBNN training by the proposed gradient based l_2 optimization approach in batch mode ranged from 15 s (for small neural network with 100 training samples) to 5 min (for large neural network with 500 training samples).

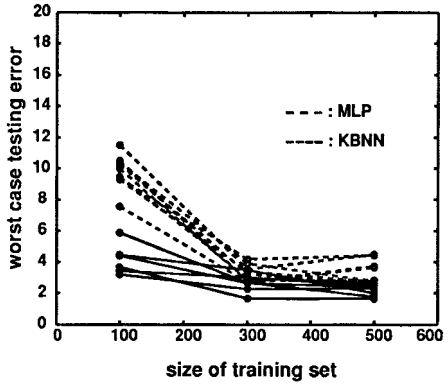


(a)

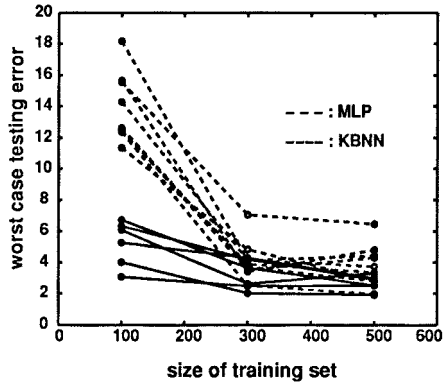


(b)

Fig. 5. Model accuracy comparison of KBNN and MLP in terms of average testing error for the transmission line example. (a) Testing data sampled within the same range as training data (b) Testing data sampled around/beyond the boundary of training data. The curves are from models of various sizes and trainings with different initial weights. The advantage of KBNN over MLP is even more siginificant when less training data is available. KBNN is also much more reliable than MLP in the extrapolation region, i.e., in case (b).



(a)



(b)

Fig. 6. Model accuracy comparison of KBNN and MLP in terms of worst case testing error for the transmission line example. (a) Testing data sampled within the same range as training data (b) Testing data sampled around/beyond the boundary of training data. The curves are from models of various sizes and trainings with different initial weights. The advantage of KBNN over MLP is even more siginificant when less training data is available. KBNN is also much more reliable than MLP in the extrapolation region, i.e., in case (b).

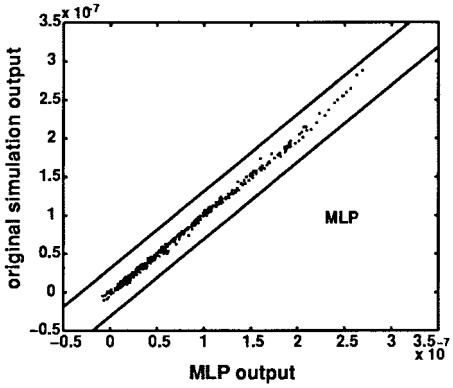
TABLE III
MODEL ACCURACY COMPARISON BETWEEN MLP AND KBNN FOR TRANSMISSION LINE MODELING EXAMPLE WITH TESTING DATA IN THE SAME REGION AS TRAINING DATA. THE RESULTS SHOWN ARE THE AVERAGE OF THREE DIFFERENT TRAININGS FOR EACH MODEL

Training sample size	Neural net type	Model size	Average test error	Largest test error
100	Standard (MLP)	7	0.95%	9.30%
		15	1.18%	10.07%
		20	1.33%	10.04%
	Knowledge based (KBNN)	b2z3	0.51%	4.18%
		b4z6	0.64%	4.16%
300	Standard (MLP)	7	0.58%	3.12%
		15	0.56%	3.29%
		20	0.58%	3.39%
	Knowledge based (KBNN)	b2z3	0.44%	2.59%
		b4z6	0.41%	2.64%
500	Standard (MLP)	7	0.51%	3.38%
		15	0.54%	3.30%
		20	0.56%	3.28%
	Knowledge based (KBNN)	b2z3	0.41%	2.02%
		b4z6	0.38%	2.19%

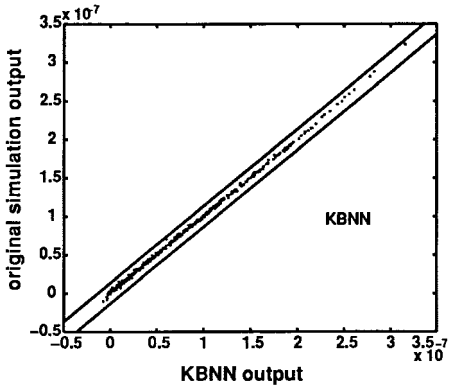
TABLE IV
MODEL ACCURACY COMPARISON BETWEEN MLP AND KBNN FOR TRANSMISSION LINE MODELING EXAMPLE WITH TESTING DATA AROUND/BEYOND TRAINING DATA BOUNDARY. THE RESULTS SHOWN ARE THE AVERAGE OF THREE DIFFERENT TRAININGS FOR EACH MODEL

Training sample size	Neural net type	Model size	Average test error	Largest test error
100	Standard (MLP)	7	2.38%	12.09%
		15	2.66%	13.57%
		20	3.09%	16.00%
	Knowledge based (KBNN)	b2z3	1.04%	5.43%
		b4z6	1.05%	5.01%
300	Standard (MLP)	7	1.01%	3.61%
		15	0.91%	3.66%
		20	0.96%	5.11%
	Knowledge based (KBNN)	b2z3	0.69%	2.95%
		b4z6	0.98%	3.47%
500	Standard (MLP)	7	0.85%	3.05%
		15	0.89%	3.65%
		20	0.91%	4.87%
	Knowledge based (KBNN)	b2z3	0.77%	2.70%
		b4z6	0.87%	2.66%

A set of 500 testing samples were generated in the same range as Table II to test the trained neural models with results shown in Table III. These testing data were never used in training. A further set of testing data with 4096 samples were deliberately selected around/beyond the boundary of the model effective region in input parameter space in order to compare extrapolation accuracy of KBNN and MLP as shown in Table IV. A significant superior performance of KBNN over MLP is demonstrated in the case with smaller training data set, e.g., 100 samples. Furthermore, the overall tendency suggests that the accuracy of KBNN trained by a small set of training

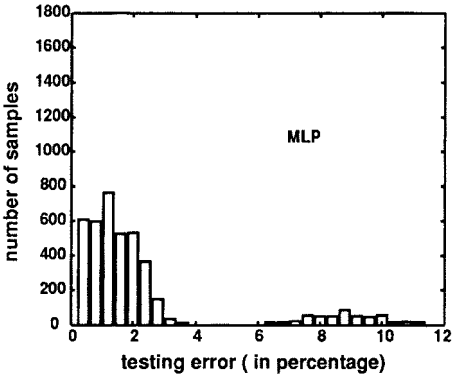


(a)

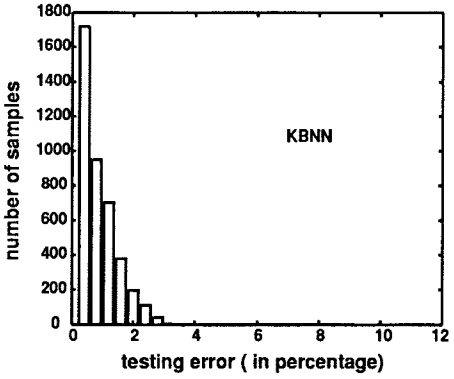


(b)

Fig. 7. Scattering plot of mutual inductance l_{12} (a) from MLP and (b) from KBNN for the transmission line modeling example for 500 testing samples. Both models were trained with insufficient training data of only 100 samples.



(a)



(b)

Fig. 8. Histograms of testing error of (a) MLP and (b) KBNN for the transmission line modeling example for 4069 testing samples around/beyond training data boundary. Both models were trained by only 100 training samples. Since concentration of errors is closer to 0% for KBNN than that of MLP, KBNN shows better accuracy than MLP.

TABLE V
TRAINING DATA RANGES OF NEURAL MODEL INPUT PARAMETERS FOR MESFET MODELING EXAMPLE

Parameters	Notation	Range
Gate Length	L	0.336 – 0.504 μm
Gate Width	W	0.8 – 1.2 mm
Channel Thickness	a	0.28 – 0.42 μm
Doping Density	N_d	$1.68 \times 10^{23} - 2.52 \times 10^{23}$ $1/m^3$
Gate Voltage	V_G	-5 – 0 V
Drain Voltage	V_D	0 – 4 V

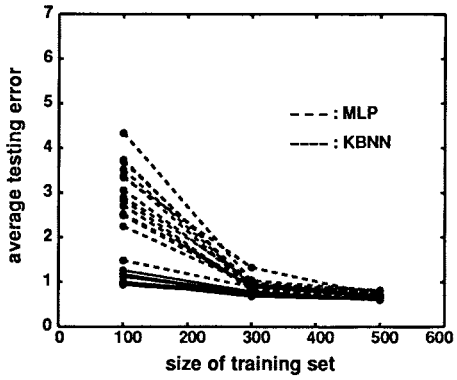
TABLE VI
EXTRAPOLATION DATA RANGES OF NEURAL MODEL INPUT PARAMETERS FOR MESFET MODELING EXAMPLE

Parameters	Notation	Range
Gate Length	L	0.315 – 0.525 μm
Gate Width	W	0.75 – 1.25 mm
Channel Thickness	a	0.263 – 0.438 μm
Doping Density	N_d	$1.58 \times 10^{23} - 2.63 \times 10^{23}$ $1/m^3$
Gate Voltage	V_G	-5 – 0 V
Drain Voltage	V_D	0 – 4 V

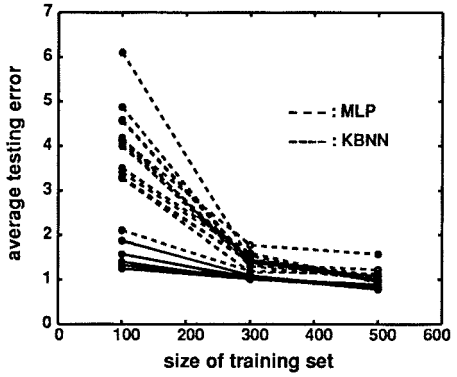
data is comparable to that of MLP trained by a larger set of training data. Figs. 5 and 6 reveal more information by showing the error from individual trainings of KBNN and MLP in terms of average and the worst case testing error, respectively. A much more stable performance of KBNN when making an extrapolation prediction is observed over MLP. The error for KBNN increases much slowly compared to that of MLP when test data moves to extrapolation region. Fig. 7 shows the scattering plots of mutual inductance between neural models (MLP with seven hidden neurons and KBNN (b2z3)) and original simulation for 500 testing samples within training data boundary. The ideal plot is that all points should be at the diagonal line. The plot for KBNN is closer to the diagonal line and has smaller worst case error envelope. Fig. 8 shows the histograms of error of MLP and KBNN for testing samples around/beyond training data boundary when trained by insufficient data of only 100 samples. The error for KBNN mostly concentrate in the vicinity of zero. MLP has some error distributed at much higher error level. This also indicates the reliability of KBNN model.

C. Example 3: MESFET Modeling

This example demonstrates the use of the proposed KBNN to model physics-based MESFET [10] and its comparison with traditional MLP. Device physical/process parameters(channel length L, channel width W, doping density N_d, channel thickness a) and terminal voltages, i.e., gate-source voltage (V_G) and drain-source voltage (V_D), are neural network input parameters and drain current, i.e., i_d, is the neural network output. The original problem is physics-based [31] and requires a slow numerical simulation procedure. The neural network models (KBNN or MLP) are much faster than the original physics-based FET model, e.g., about 4 s by KBNN/MLP and 27 min by original FET model to do 1000 repetitive



(a)



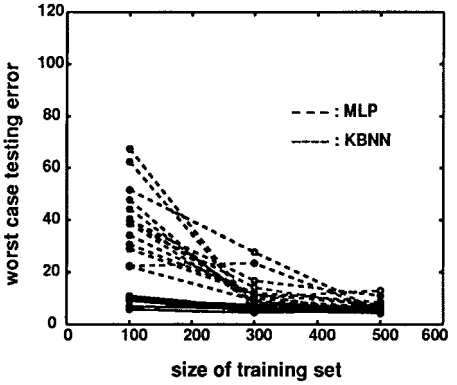
(b)

Fig. 9. Model accuracy comparison of KBNN and MLP in terms of average testing error for the MESFET example. (a) Testing data sampled within the same range as training data. (b) Testing data sampled out of the boundary of training data by 25% as shown in Table VI. The curves are from models of various sizes and trainings with different initial weights. The advantage of KBNN over MLP is even more siginificant when less training data is available. KBNN is also much more reliable than MLP in the extrapolation region, i.e., in case (b).

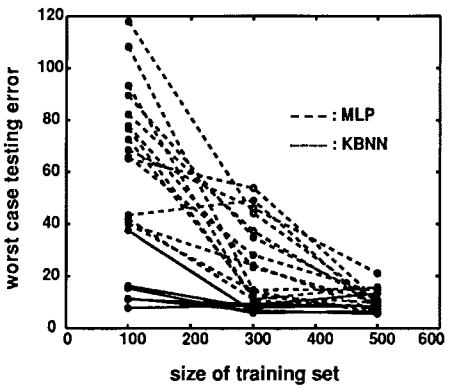
TABLE VII
MODEL ACCURACY COMPARISON BETWEEN STANDARD MLP AND KBNN FOR MESFET MODELING EXAMPLE WITH TESTING DATA FROM THE SAME REGION AS TRAINING DATA. THE RESULTS SHOWN ARE THE AVERAGE OF THREE DIFFERENT TRAININGS FOR EACH MODEL

Training sample size	Neural net type	Model size	Average test error	Largest test error
100	Standard (MLP)	7	2.14%	45.20%
		10	2.91%	30.80%
		14	3.38%	52.53%
		18	2.75%	35.05%
		25	3.78%	39.71%
	Knowledge based (KBNN)	b5z6	1.12%	8.99%
		b6z8	1.03%	8.49%
300	Standard (MLP)	7	0.90%	8.74%
		10	0.90%	12.89%
		14	0.96%	14.16%
		18	0.89%	11.09%
		25	1.11%	13.66%
	Knowledge based (KBNN)	b5z6	0.74%	5.68%
		b6z8	0.72%	5.72%
500	Standard (MLP)	7	0.74%	6.15%
		10	0.70%	5.39%
		14	0.68%	6.59%
		18	0.69%	7.51%
		25	0.78%	10.08%
	Knowledge based (KBNN)	b5z6	0.61%	5.37%
		b6z8	0.61%	5.47%

simulations in a Monte Carlo analysis with random values of device physical/geometrical parameters.



(a)



(b)

Fig. 10. Model accuracy comparison of KBNN and MLP in terms of worst case testing error for the MESFET example. (a) Testing data sampled within the same range as training data. (b) Testing data sampled out of the boundary of training data by 25% as shown in Table VI. The curves are from models of various sizes and trainings with different initial weights. The advantage of KBNN over MLP is even more siginificant when less training data is available. KBNN is also much more reliable than MLP in the extrapolation region, i.e., in case (b).

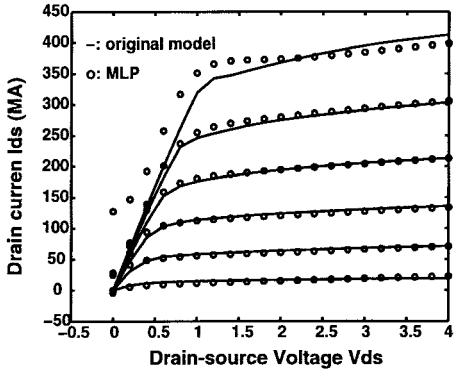
There exist empirical formulas for MESFET modeling, e.g., [23]. The KBNN is developed incorporating empirical formulas in knowledge layer $\mathbf{Z}$. Training samples were first obtained by simulating original Khatibzadeh and Trew models [31] using OSA90[1] at randomly selected points. The data range is shown in Table V. Three sets of training data with 100, 300, and 500 samples, respectively, were used. The neural net training was done on SPARC station 5. The CPU time for MLP training by the conventional sample-by-sample error backpropagation approach ranged from 22 to 60 min. The CPU time for MLP or KBNN training by proposed gradient based l_2 optimization approach in batch mode ranged from 20 s to 9 min.

The ability to extrapolate beyond the boundary of training data is a challenge but an important aspect of a model. Two sets of testing data, one in the same region as training data in input parameter space and the other is out of the region by 25% (i.e., extrapolation region) shown in Table VI, were used to test neural network models of various sizes. The results are tabulated in Tables VII and VIII, respectively. In both cases, KBNN outperforms MLP in all the accuracy measures. The

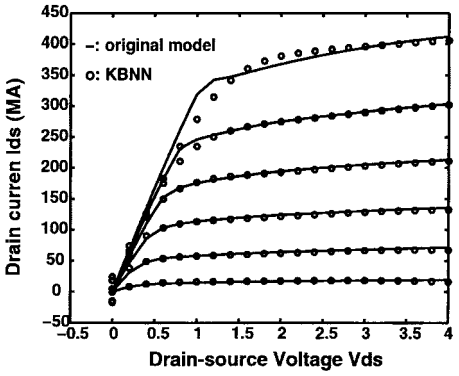
[1] *OSA90 Vesion 3.0*, Optimization Systems Associations Inc., Dundas, ON, Canada.

TABLE VIII
MODEL ACCURACY COMPARISON BETWEEN STANDARD MLP AND KBNN FOR MESFET MODELING EXAMPLE WITH TESTING DATA IN THE EXTRAPOLATION REGION. THE RESULTS SHOWN ARE THE AVERAGE OF THREE DIFFERENT TRAININGS FOR EACH MODEL.

Training sample size	Neural net type	Model size	Average test error	Largest test error
100	Standard (MLP)	7	2.88%	51.52%
		10	3.86%	53.38%
		14	4.22%	96.31%
		18	3.82%	71.04%
		25	5.04%	88.70%
	Knowledge based (KBNN)	b5z6	1.56%	21.69%
		b6z8	1.43%	11.31%
300	Standard (MLP)	7	1.24%	12.67%
		10	1.43%	28.31%
		14	1.47%	25.91%
		18	1.38%	39.71%
		25	1.52%	31.83%
	Knowledge based (KBNN)	b5z6	1.04%	6.96%
		b6z8	1.02%	7.92%
500	Standard (MLP)	7	1.10%	12.67%
		10	1.00%	7.64%
		14	0.96%	9.35%
		18	0.99%	10.86%
		25	1.29%	15.53%
	Knowledge based (KBNN)	b5z6	0.83%	6.64%
		b6z8	0.83%	9.40%

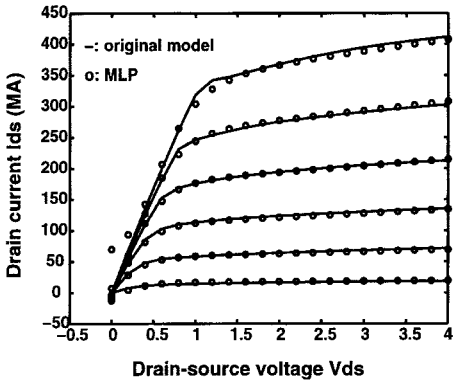


(a)

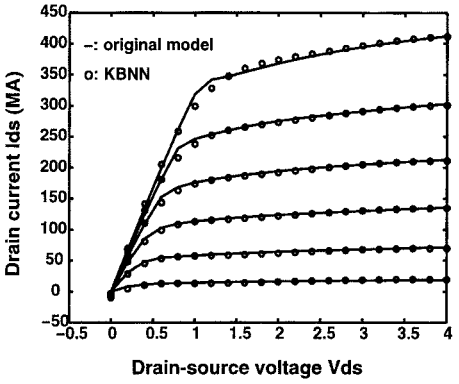


(b)

Fig. 11. An example of IV curves from (a) MLP and (b) KBNN for MESFET modeling example. Both models were trained with insufficient training data of only 100 samples. The 100 samples were generated by changing 6 FET parameters including gate width, length, channel thickness, dopping density, $V_{\rm GS}$ and $V_{\rm DS}$. KBNN is visiblely better than standard MLP.



(a)



(b)

Fig. 12. An example of IV curves from (a) MLP and (b) KBNN for MESFET modeling example. Both models were trained with reasonable size of training data of 300 samples.

superiority is even more significant when fewer training data is available. The overall tendency is that KBNN trained with 100 samples can achieve similar accuracy as that of MLP trained with 300 samples. And KBNN trained by 300 samples is as accurate as MLP trained by 500 samples. Figs. 9 and 10 reveal more information by showing the errors from individual trainings. All the trained KBNN's perform better than any trained MLP's when training data set is small.

Moving to the extrapolation region, the accuracy of KBNN's deteriorates much more slowly than that of MLP's. This is because the built-in knowledge in the KBNN gives it more information not seen in the training data. Fig. 11 shows an example of IV curves from the best performing MLP (with 7 hidden neurons) and KBNN (b5z6) models, both trained by insufficient training data of 100 samples. KBNN is visibly better than MLP. Fig. 12 shows an example of IV curves from the same models when trained by 300 samples.

V. CONCLUSION

A KBNN has been proposed combining microwave empirical experience with the power of learning of neural networks. A new error backpropagation training scheme for the KBNN structure utilizing gradient based l_2 optimization has been developed. For the examples presented in this paper, the model testing errors from KBNN are less than that from MLP. The advantage of KBNN is even more significant when training data is insufficient. Reductions in the cost of model development through reduced need of generating large amount of training data and more efficient training algorithm have

been demonstrated. The neural models can learn and predict component behaviors originally seen in detailed physics/EM models, and predict such behavior much faster than original models. This work is significant for the growing use of neural networks as economical and accurate models in microwave design. It will have a significant impact on statistical analysis and design of microwave circuits.

REFERENCES

[1] Q. J. Zhang and M. S. Nakhla, "Signal integrity analysis and optimization of VLSI interconnects using neural network models," in *IEEE Int. Symp. Circuits Systems*, London, U.K., 1994, pp. 459–462.

[2] 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.

[3] 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.

[4] G. L. Creech, B. Paul, C. Lesniak, T. Jenkins, R. Lee, and M. Calcatera, "Artificial neural networks for accurate microwave CAD applications," in *IEEE Int. Microwave Symp. Digest*, San Francisco, CA, 1996, pp. 733–736.

[5] V. B. Litovski, J. Radjenovic, Z. M. Mrcarica, and S. L. Milenkovic, "MOS transistor modeling using neural network," *Electron. Lett.*, vol. 28, no. 18, pp. 1766–1768, 1992.

[6] A. H. Zaabab, Q. J. Zhang, and M. S. Nakhla, "Analysis and optimization of microwave circuits and devices using neural network models," in *IEEE Int. Microwave Symp. Dig.*, San Diego, CA, May 1994, pp. 393–396.

[7] T. Horng, C. Wang, and N. G. Alexopoulos, "Microstrip circuit design using neural networks," in *IEEE Int. Microwave Symp. Dig.*, Atlanta, Georgia, 1993, pp. 413–416.

[8] M. Vai and S. Prasad, "Automatic impedance matching with a neural network," *IEEE Microwave Guided Wave Lett.*, vol. 3, pp. 353–354, 1993.

[9] 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.

[10] A. H. Zaabab, Q. J. Zhang, and M. S. Nakhla, "Neural network modeling approach to circuit optimization and statistical design," *IEEE Trans. Microwave Theory Tech.*, vol. 43, pp. 1349–1358, 1995.

[11] R. Biernacki, J. W. Bandler, J. Song, and Q. J. Zhang, "Efficient quadratic approximation for statistical design," *IEEE Trans. Circuits Syst.*, vol. 36, pp. 1449–1454, 1989.

[12] P. Meijer, "Fast and smooth highly nonlinear multidimensional table models for device modeling," *IEEE Trans. Circuits Syst.*, vol. 37, pp. 335–346, 1990.

[13] Q. J. Zhang, F. Wang, and M. S. Nakhla, "Optimization of high-speed VLSI interconnects: A review," *Int. J. Microwave and Millimeter-Wave CAD*, vol. 7, pp. 83–107, 1997.

[14] S. Haykin, *Neural Networks*. New York, IEEE, 1994.

[15] S. W. Piche, "Steepest descent algorithms for neural network controllers and filters," *IEEE Trans. Neural Networks*, vol. 5, pp. 198–212, Mar. 1994.

[16] R. P. Lippmann, "Review of neural networks for speech recognition," *Neural Computation*, vol. 1, pp. 1–38, 1989.

[17] R. Reed, "Pruning algorithms—A survey," *IEEE Trans. Neural Networks*, vol. 4, pp. 740–747, Sept. 1993.

[18] G. G. Towell and J. W. Shavlik, "Knowledge-based artificial neural networks," *Artif. Intell.*, vol. 70, pp. 119–165, 1994.

[19] L. M. Fu, "Integration of neural heuristics into knowledge-based inference," *Connection Sci.*, vol. 1, pp. 325–340, 1989.

[20] R. C. Lacher, S. I. Hruska, and D. C. Kuncicky, "Back-propagation learning in expert networks," *IEEE Trans. Neural Networks*, vol. 3, pp. 62–72, Jan. 1992.

[21] J. J. Mahoney and R. J. Mooney, "Combining connectionist and symbolic learning to refine certainty factor rule bases," *Connection Sci.*, vol. 5, nos. 3 and 4, pp. 339–364, 1993.

[22] C. S. Walker, *Capacitance, Inductance and Crosstalk Analysis*. Boston, MA: Artech House, 1990.

[23] P. H. Ladbrooke, *MMIC Design: GaAs FET's and HEMTs*. Boston, MA: Artech House, 1989.

[24] F. Wang and Q. J. Zhang, "Knowledge based neural models for microwave design," in *IEEE Int. Microwave Symp. Dig.*, Denver, CO, June 1997, vol. II, pp. 627–630.

[25] H. Leung and S. Haykin, "Rational function neural network," *Neural Computation*, vol. 5, pp. 928–938, 1993.

[26] A. Abdelbar and G. Tagliarini, "Honest: A new high order feedforward neural networks," in *Proc. IEEE Int. Conf. Neural Networks*, Washington, DC, June 1996, pp. 1257–1262.

[27] J. R. Tsai, P. C. Chung, and C. I. Chang, "A sigmoidal radial basis function neural network for function approximation," in *Proc. IEEE Int. Conf. Neural Networks*, Washington, DC, June 1996, pp. 496–501.

[28] J. W. Bandler and Q. J. Zhang, "Optimization techniques for modeling, diagnosis and tuning," in *Analog Methods for Computer-Aided Circuit Analysis and Diagnosis*, T. Ozawa, Ed. New York: Marcel Dekker, 1988, ch. 14.

[29] D. Zhou, S. Su, F. Tsui, D. S. Gao, and J. Cong, "A simplified synthesis of transmission lines with a tree structure," *Analog Integrated Circuits and Signal Processing*, vol. 5, pp. 19–30, 1994.

[30] R. F. Harrington, *Field Computation by Moment Methods*. New York: Macmillan, 1968.

[31] M. A. Khatibzadeh and R. J. Trew, "A large-signal, analytical model for the GaAs MESFET," *IEEE Trans. Microwave Theory Tech.*, vol. 36, no. 2, pp. 231–239, 1988.

Fang Wang (S'94) received the B.Eng. degree from Xi'an Jiaotong University, Xi'an, China, in 1993, and the M.Eng. degree from Carleton University, Ottawa, ON, Canada, in 1995, both in electrical engineering, and is currently completing the Ph.D. degree in Department of Electronics at Carleton University, Ottawa, ON, Canada.

She is a recipient of a 1997–1999 Postgraduate Scholarship from the Natural Science and Engineering Research Council of Canada (NSERC). She is also a two-time recipent of Ontario Graduate Scholarship in 1996 and 1997, respectively. Her research interests include neural networks, modeling and their applications in CAD for electronics circuits.

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

He was with the Institute of Systems Engineering, Tianjin University, Tianjin, China, from 1982 to 1983. He was a research engineer with Optimization Systems Associates Inc., Dundas, ON, Canada, from 1988 to 1990. During 1989 and 1990, he was also an Assistant Professor (part-time) of Electrical and Computer Engineering in McMaster University. He joined the Department of Electronics, Carleton University, Ottawa, Canada, in 1990 where he is presently an Associate Professor. His research interests are computer-aided design of high-speed/high-frequency circuits, with emphasis on optimization, neural networks, modeling, simulation, and statistical design of microwave circuits and high-speed VLSI interconnections. He has over 100 publications in the area in international journals and conferences. He is a co-editor of *Modeling and Simulation of High-Speed VLSI Interconnects* (Kluwer, 1994) and a contributor to *Analog Methods for Computer-Aided Analysis and Diagnosis*, (Marcel Dekker, 1988).