Sensitivity-Analysis-Based Adjoint Neural Network Techniques for Nonlinear Applications

by

Sayed Alireza Sadrossadat, M.A.Sc.

A thesis submitted to

the Faculty of Graduate and Postdoctoral Affairs

in partial fulfillment of the degree requirements of

Doctor of Philosophy

Ottawa-Carleton Institute for

Electrical and Computer Engineering

Department of Electronics

Carleton University

Ottawa, Ontario, Canada, 2015

Abstract

Artificial neural networks (ANN) have recently emerged as a powerful computer-aided design (CAD) tool for modeling nonlinear devices and circuits. The overall objective of this thesis is to develop sensitivity analysis based neural network techniques for both frequency domain and transient modeling of nonlinear circuits. The proposed techniques not only adds sensitivity data to the obtained model but also makes conventional training more efficient. The first contribution of this thesis is the development of sensitivity-analysis-based adjoint neural-network (SAANN) technique for modeling microwave passive components. This method adds sensitivity data to the obtained model. In addition, the SAANN technique reduces the amount of training data required for model development increasing the efficiency of model development. As a further contribution, this thesis presents a novel robust modeling technique, adjoint state-space dynamic neural network (ASSDNN), for transient modeling of nonlinear optical/electrical components and circuits. This technique adds time-domain sensitivity data, which does not exist in current opto-electronic and physics-based simulators, to the output of the obtained model. The proposed technique requires less training data for creating the model and consequently makes training faster and more efficient. Furthermore, this technique was developed such that it can take advantage of parallel computation. This results in the technique being much faster and efficient than conventional transient modeling techniques. In addition, the evaluation time for models of nonlinear optical-electrical and physics-based devices generated using the proposed technique is reduced compared to current simulation tools.

To my parents

Fatemeh Pourmoghadas and Hamid Sadrossadat

Acknowledgments

First of all, I would like to thank my supervisors, Professor Pavan Gunupudi and Professor Qi-Jun Zhang for their constant support, encouragement, guidance and expert supervision throughout my PhD's program. Their professional leaderships have made the research through my PhD program a rewarding journey. Their continuous striving for research at the highest level will influence me for my professional future life. It was my honor to work under their supervision and guidance.

Specially, I'd like to thank Dr. Michel Nakhla, Dr. Roni Khazaka, Dr. Ram Achar, Dr. Peter Liu, and Dr. Rony Amaya as the readers of my thesis, for their invaluable suggestions and corrections and also Dr. Yazi Cao who helped me in the first part of my research.

Finally, I'd like to thank my parents, the two most influential persons in my life. Without their continuous support, love and encouragement the accomplishment of this thesis would not have been possible for me. They have given me tremendous strength to go through the difficulties of my PhD program. I dedicate this thesis to them.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Introduction to Artificial Neural Networks

The fast development of commercial markets for wireless communication products in recent years has led to an increasing interest for improving circuit design methods in the radio frequency (RF) and microwave topics. The older discipline of Department of Defense (DOD)-oriented RF/microwave electronics with the emphasis on performance is being replaced by this new market for the expertise in high frequency after the defense build-down in the early 1990s. Modern wireless communication systems need an understanding of RF and microwave circuit design methods and a background in digital communication methods and also knowledge about the current and emerging standards of the wireless communication protocol. The emphasis of the wireless industry on low cost and time-to-market, are creating increasing demands on computer-aided design (CAD) tools for RF/microwave circuits, and electrical/optical systems.

Electromagnetic (EM) simulation methods for high frequency structures devel-

oped recently brought the CAD for RF/microwave circuits to its current state of the art. The use of trained artificial neural network (ANN) models by EM simulators [1]-[3] is among the recent developments that led to an efficient usage of EM simulation for RF and microwave CAD. In this way, EM simulation calculates S-parameters for all the components to be modeled over the ranges that they are going to be used. Using the obtained data from EM simulations to train ANN leads to an ANN model for all of the components. ANNs can also be used for modeling active devices and for circuit optimization and statistical design [4].

Generally ANNs are strong techniques for modeling any input/output relationships. Many applications have been reported in several areas such as control [5], telecommunications [6], biomedical [7], remote sensing [8], pattern recognition [9] , and manufacturing [10]. However, ANNs are being used frequently in the RF/microwave design area [11]. Several applications are also reported in automatic impedance matching [12], microstrip circuit design [13], microwave circuit analysis and optimization [14], [15], active device modeling [15]-[17], modeling of passive components [1]-[3], [18],[19], and modeling for electro/opto interconnections [20]. Several other advanced works have been done in RF and microwave-oriented neural network structures [21]-[22], training algorithms [22]-[23], white-box modeling [24], and knowledge based networks [19], [25]-[26], [27].

In the circuit simulation area, ANNs can be used to develop models using input-output data obtained from components that can replace traditional models leading to faster execution times with good accuracy. The growing complexity of high frequency nonlinear circuits has brought out a need to develop faster models capturing

the dynamic behavior of nonlinear components and systems [86]-[94]. Attempts to address this need started with the introduction of discrete-time recurrent neural networks [86], [87], [92], [94], [95]. There are several other structures of neural networks such as multilayer perceptron neural networks (MLP) [96], real valued time-delay neural networks [90], radial basis function neural networks ([97], [91]), continuous-time dynamic neural networks ([88], [89]), and the recently introduced state-space dynamic neural networks (SSDNN) [98]-[102]. SSDNN can be seen as a generalized form of DNN-based techniques [88], [98].

In this thesis, two new training techniques for static and dynamic neural network are developed. The first developed method, sensitivity-analysis-based adjoint neural-network technique (SAANN), is an advance over the conventional static MLP which adds the sensitivity information to the conventional ANN by formulating new backward-forward propagation technique and it can be incorporated into the existing CAD tools. Using this method, not only adds the sensitivity information beyond the variable limits of CAD tools but also makes the training more efficient requiring less training data and leading to less model development cost. Several examples have been demonstrated to verify the validity of the proposed method in this thesis.

The second developed technique, adjoint state-space dynamic neural network (ASSDNN), is an advance over the conventional state-space dynamic neural network for transient modeling of nonlinear circuits/components. The proposed ASSDNN, similar to the proposed SAANN for static ANN structures, adds the sensitivity information to the conventional dynamic state-space models which leads to less training data being required for training. Therefore, the proposed method, providing

the sensitivity information, not only makes the evaluation time much faster than traditional simulation tools, but also makes the training more efficient compared to the conventional dynamic state-space neural network techniques. Also, for the first time, this training algorithm was formulated such that it can be processed using parallel cores which makes a significant improvement in the training process. Several optical/electrical examples have been presented to prove the validity of the proposed technique.

1.2 List of Contributions

1.2.1 Sensitivity-Analysis-Based Adjoint Neural-Network Technique [103]

The following contributions were made in the development of the new adjoint sensitivity-analysis-based technique (SAANN) for modeling microwave components:

- Formulating a new error function based on the conventional error and the sensitivity errors.
- Developing new recursive forward-backward propagation method to obtain the second derivative information.
- Deriving the gradient of the objective function including the sensitivities using the new formula for second derivative information in order to develop the new training method.

1.2.2 Adjoint State-Space Dynamic Neural Network Technique for Nonlinear Transient Modeling [99]

the following contributions were made in the development of the new adjoint state-space dynamic neural network technique (ASSDNN) for modeling optical/electrical components and circuits:

- Formulating the new adjoint system based on the original response of the system and their sensitivities.
- Formulating a novel constrained optimization problem using Lagrangian functions to train models developed using the ASSDNN technique.
- Proof that the proposed technique produces lower training error compared to traditional training techniques that do not use sensitivity information.
- Proof that the new system obtained through the proposed method is stable.
- Formulation of the proposed method to run on parallel cores.

1.3 Thesis Organization

The rest of the thesis is organized as follows: Chapter 2 presents an overview of ANN-based techniques as well as dynamic-ANN methods. Chapter 3 presents a new sensitivity-analysis-based adjoint neural network technique that is an advance over conventional neural network techniques. This is followed by Chapter 4 which presents a new adjoint state-space dynamic neural network technique for modeling

nonlinear components in the time-domain. Finally Chapter 5 presents the conclusions and discusses future work that can be performed in this area.

Chapter 2

Literature Review

ANNs have several structures that will be discussed in the next section. Regardless of the structure of ANNs, all of them have at least two physical components, the processing elements and the connections between them. The processing elements are called neurons, and the connections between them are called links. There is a weight parameter associated with each link. Each neuron receives outputs coming from the neurons connected to it, processes the information, and produces an output. Input neurons are the neurons that receive information from outside the network (i.e., not from neurons of the network) and the output neurons are ones whose outputs are used externally. Hidden neurons are the neurons that receive information from other neurons and pass the processed information to the other neurons in the neural network. There are several ways of processing information by a neuron, and several ways of connecting the neurons to each another. Therefore, by using different processing elements and by the different ways of connection between them, several neural network structures can be created.

Different structures of neural network have been developed so far for signal

processing, pattern recognition, control and so on. In the next section, the most commonly used structures of neural networks will be described [1], [28]. These structures include multilayer perceptrons (MLP), radial basis function networks (RBF), and recurrent neural networks (RNN).

2.1 Neural Network Structures

Assume N_x and N_y represent the number of input and output neurons of the neural network, x be an N_x-vector including the external inputs to the neural network, y be an N_y-vector including the outputs from the output neurons, and w be a vector including all the weight parameters representing the connections in the neural network. The function $y = y(x, w)$ mathematically represents a neural network. The definition of w and the way that y is calculated from x and w, determine the structure of the neural network.

2.1.1 Multilayer Perceptions (MLP) Networks

MLPs are the most popular type of neural networks being used in many different applications. They are part of a general class of structures called feedforward neural networks [29]. MLP neural networks have been used in several modeling and optimization problems.

In the structure of MLP, the neurons are grouped into different layers. The first and last layers are called input and output layers, respectively. The rest of the layers are called hidden layers. Typically, an MLP includes one input layer, one or more hidden layers, and one output layer, as shown in Figure 2.1. Suppose L as the

total number of layers. The first layer is the input layer, the Lth layer is the output layer, and layers 2 to $L-1$ are hidden layers. Suppose the number of neurons in l^{th} layer to be $N_l, l = 2, ..., L$.

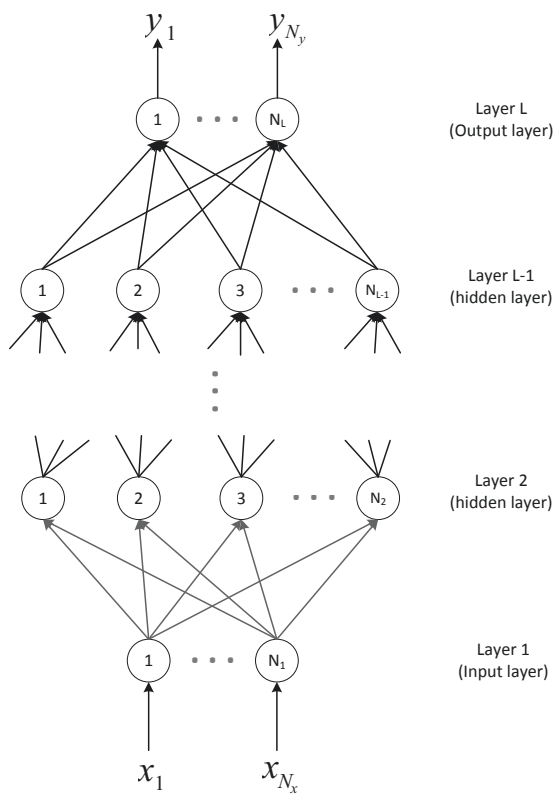


Figure 2.1: Multilayer perceptrons (MLP) structure containing one input layer, one output layer, and several hidden neurons.

Suppose w_{ij}^l represents the weight of the link between j^{th} neuron of $(l-1)^{th}$ layer and i^{th} neuron of lth layer (for $1 \leq j \leq N_{l-1}, 1 \leq i \leq N_l$). Assume x_i to be the ith input to the MLP, and z_i^l be the output of ith neuron of lth layer. Also, let w_{i0}^l represents the bias for i^{th} neuron of lth layer. Therefore, the vector of weights

in MLP is,

$$w = [w_{10}^2\, w_{11}^2\, w_{12}^2\, \cdots w_{1N_1}^2 w_{10}^3 \cdots w_{N_L N_{L-1}}^L]^T \tag{2.1}$$

Let $\sigma(.)$ be the activation function of a hidden neuron in MLPs. There are several activation functions for hidden neurons. The sigmoid function is the most common one as follows,

$$\sigma(\gamma) = \frac{1}{1 + e^{-\gamma}} \tag{2.2}$$

Other possible activation functions are arc-tangent function,

$$\sigma(\gamma) = \left(\frac{2}{\pi}\right) \arctan(\gamma) \tag{2.3}$$

and hyperbolic-tangent function,

$$\sigma(\gamma) = \frac{e^{\gamma} - e^{-\gamma}}{e^{\gamma} + e^{-\gamma}} \tag{2.4}$$

All of these functions are bounded, continuous, monotonic, and continuously differentiable. Also, the linear activation function that is used to calculate MLP output is defined as,

$$\sigma(\gamma_i^L) = \gamma_i^L = \sum_{j=0}^{N_{L-1}} w_{ij}^L z_j^{L-1} \tag{2.5}$$

Now the feedforward process is to pass the external inputs to the input neurons, then the outputs from the input neurons are fed to the hidden neurons of the 2nd layer, and so on, and finally the outputs of $(L-1)^{th}$ layer are fed to the output

neurons (the last layer). This process can be summarized as,

$$z_i^1 = x_i,\ i = 1, 2, \cdots, N_1,\ N_1 = N_x \tag{2.6}$$

$$z_i^l = \sigma\left(\sum_{j=0}^{N_{l-1}} w_{ij}^l z_j^{l-1}\right),\ i = 1, 2, \cdots, N_l,\ l = 2, 3, \cdots, L \tag{2.7}$$

$$y_i = z_i^L,\ i = 1, 2, \cdots, N_L,\ N_L = N_y \tag{2.8}$$

According to the universal approximation theorem for MLP that was proved by Cybenko [30] and Hornik et al. [31], a three layer perceptron (a perceptron is an algorithm for supervised learning) provided by enough hidden neurons, is capable of approximating an arbitrary nonlinear, continuous, multi-dimensional function with any desired accuracy. Practically, the exact number of hidden neurons required for a modeling task is still an open question. The ongoing research in this direction includes methods such as constructive algorithms [32], network pruning [33], and regularization [34], to match the complexity of the neural network model with complexity of the problem.

In practice, three-layer or four-layer perceptrons are most commonly used for many applications. Perceptively, a four-layer perceptrons would perform better in modeling nonlinear problems whereas a three-layer perceptron neural network-although capable of modeling such problems-may need too many hidden neurons to realize the same behavior. In the function approximation where generalization capability is a major concern, three-layer perceptrons are usually preferred [35], because

the resulting network usually includes fewer hidden neurons. It was demonstrated in [36] that four-layer perceptrons have better performance in boundary definitions so they are usually preferred in pattern classification problems.

The purpose in neural model development is to find the optimum weight parameters w, such that $y = y(x, w)$ is similar to the formulation of the original problem. This goal is achieved through a process called training. The training data that is passed to the neural network include pairs of (x_k, d_k), $k \in I$, where d_k is the desired outputs of the neural model for inputs x_k, and I is the set of training samples.

The error function defined as the difference between the actual and the desired outputs is calculated as,

$$E\left(x, w\right) = \frac{1}{2} \sum_{k \in I} \sum_{j=1}^{N_L} \left(y_j\left(x_k, w\right) - d_{jk}\right)^2 \tag{2.9}$$

where d_{jk} is the jth element of d_k and $y_j(x_k, w)$ is the jth output of neural network for input x_k. The weight parameters w should be adjusted during training such that the error function is minimized. Rumelhart, Hinton, and Williams in 1986 [37] proposed a systematic method for training of neural network in a process called back propagation (BP) algorithm. In the following, it's explained how to compute the gradient information, $\frac{\partial E_k}{\partial w}$, in the BP algorithm.

The derivative of E_k with respect to the weight parameters of the lth layer can be calculated as,

$$\frac{\partial E_k}{\partial w_{ij}^l} = \frac{\partial E_k}{\partial z_i^l} \times \frac{\partial z_i^l}{\partial w_{ij}^l} \tag{2.10}$$

and

$$\frac{\partial z_i^l}{\partial w_{ij}^l} = \frac{\partial \sigma}{\partial \gamma_i^l} \times z_j^{l-1} \tag{2.11}$$

The gradient $\frac{\partial E_k}{\partial z_i^L}$ can be initialized at the output layer as,

$$\frac{\partial E_k}{\partial z_i^L} = y_i\left(x_k, w\right) - d_{ik} \tag{2.12}$$

and by back-propagating this error from $(l+1)^{th}$ layer to l^{th} layer the derivatives are calculated as,

$$\frac{\partial E_k}{\partial z_i^l} = \sum_{j=1}^{N_{l+1}} \frac{\partial E_k}{\partial z_j^{l+1}} \times \frac{\partial z_j^{l+1}}{\partial z_i^l} \tag{2.13}$$

As an example, if sigmoid is used as activation function of hidden neuron,

$$\frac{\partial \sigma}{\partial \gamma} = \sigma(\gamma)\left(1 - \sigma(\gamma)\right) \tag{2.14}$$

$$\frac{\partial z_i^l}{\partial w_{ij}^l} = z_i^l\left(1 - z_i^l\right) z_j^{l-1} \tag{2.15}$$

and

$$\frac{\partial z_i^l}{\partial z_j^{l-1}} = z_i^l\left(1 - z_i^l\right) w_{ij}^l \tag{2.16}$$

Let δ_i^l be defined as $\delta_i^l = \frac{\partial E_k}{\partial \gamma_i^l}$ representing local gradient at i^{th} neuron of lth layer which is given by,

$$\delta_i^L = y_i\left(x_k, w\right) - d_{ik} \tag{2.17}$$

$$\delta_i^l = \left(\sum_{j=1}^{N_{l+1}} \delta_j^{l+1} w_{ji}^{l+1}\right) z_i^l\left(1 - z_i^l\right),\ l = L, L-1, \cdots, 2 \tag{2.18}$$

and finally derivatives of the error with respect to the weights are

$$\frac{\partial E_k}{\partial w_{ij}^l} = \delta_i^l z_j^{l-1},\ l = L, L-1, \cdots, 2 \tag{2.19}$$

2.1.2 Radial Basis Function Networks (RBF) Networks

Radial basis function (RBF) neural networks are the feedforward neural networks that have a single hidden layer and use radial basis activation functions for hidden neurons. They have been applied to various applications including microwave transistors [38] and high speed I/O port of integrated circuits [39], [40].

A typical RBF neural network is shown in Figure 2.2. It includes one input layer, one radial basis hidden layer, and one output layer. Parameters c and λ in the figure are centers and standard deviations of radial basis activation functions. The Gaussian and multiquadratic are the most common radial basis activation functions in RBFs. The Gaussian function is given by,

$$\sigma(\gamma) = \exp\left(-\gamma^2\right) \tag{2.20}$$

and multiquadratic function is given by,

$$\sigma(\gamma) = \frac{1}{\left(\beta^2 + \gamma^2\right)^\alpha},\ \alpha > 0 \tag{2.21}$$

where β is a constant. Given the external inputs x, the input to the ith hidden neuron γ_i is given by

$$\gamma_i = \sqrt{\sum_{j=1}^{N_1} \left(\frac{x_j - c_{ij}}{\lambda_{ij}}\right)^2},\ i = 1, 2, \cdots N_2 \tag{2.22}$$

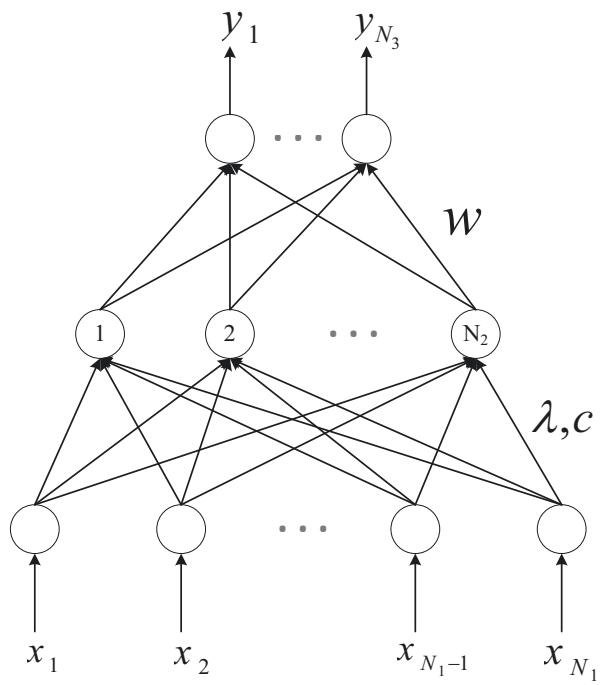


Figure 2.2: RBF neural network structure.

where N_2 is the number of hidden neurons. The output value of the ith hidden neuron assumed to be $z_i = \sigma(\gamma_i)$, where $\sigma(\gamma)$ is an RBF. Finally, the outputs of the network are calculated as

$$y_k = \sum_{i=0}^{N_2} w_{ki} z_i,\ k = 1, 2, \cdots N_3 \tag{2.23}$$

where w_{ki} is the weight of the link between i^{th} hidden neuron and k^{th} output neuron. The trainable parameters w of the RBF network consist w_{k0}, w_{ki}, c_{ij}, and λ_{ij}, where $k = 1, 2, , N_3$, $i = 1, 2, , N_2$, and $j = 1, 2, , N_1$.

2.1.3 Time Domain ANNs

In this section, two specific types of neural network structures, recurrent neural networks (RNN) and dynamic neural networks (DNN), that permit modeling of time-domain behaviors of a dynamic system are described.

Recurrent Neural Networks

In recurrent neural networks (RNN), the system outputs depend on current states of inputs and also on the history of system states and inputs [28], [41]-[43]. A typical RNN is shown in Figure 2.3. Assume history of the RNN outputs to be $y(t-\tau), y(t-2\tau), \cdots, y(t-m\tau)$ and similarly, the history of the inputs to be demonstrated as $x(t-\tau), x(t-2\tau), \cdots, x(t-n\tau)$ where m and n are the maximum number of delay steps for y and x, respectively. The system formulation can be demonstrated by,

$$y(t) = f\left(y(t-\tau), y(t-2\tau), \cdots, y(t-m\tau), x(t), x(t-\tau), x(t-2\tau), \cdots, x(t-n\tau)\right) \tag{2.24}$$

The Hopfield network is a specific type of RNN structure [44] that has a single layer. Assume it has H neurons and the neuron i can receive information from input x_i, the outputs of other neurons y_j, $j = 1, 2, ..., H$, and also from the output of the neuron itself (y_i). The output of each neuron is an external output of the neural network. Then, the activation function input of neuron i is

$$\gamma_i(t) = \sum_{j=1}^{H} w_{ij} y_j(t) + x_i(t), \quad j \neq i \tag{2.25}$$

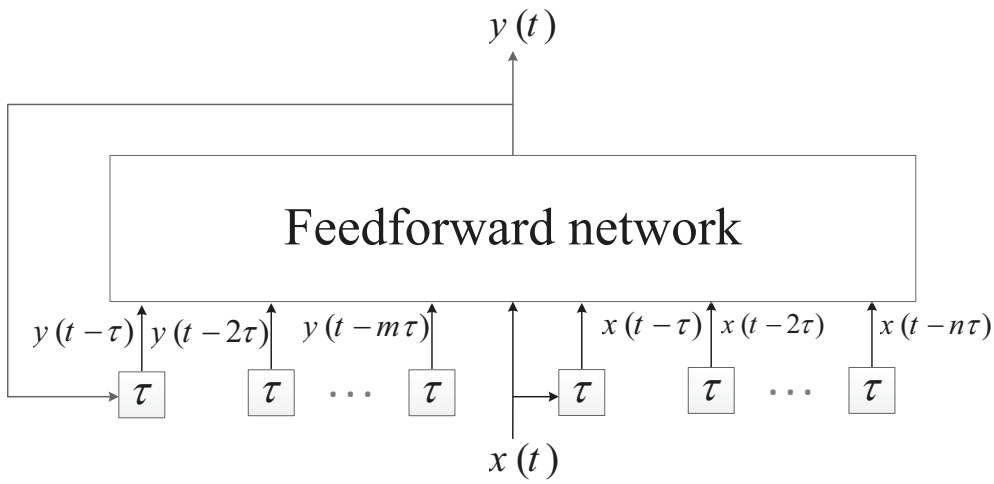


Figure 2.3: A recurrent neural network structure.

and the output of the neuron i is

$$y_i(t) = \sigma(\gamma_i(t)) \tag{2.26}$$

Dynamic Neural Networks

For best describing the nonlinear behavior of the circuits in circuit simulation, the differential dynamic neural network (DNN) was presented [45] for large signal modeling of nonlinear circuits.

Generally, the original nonlinear circuit can be described in state variable form as

$$\begin{aligned} \dot{x}(t) &= \varphi\left(x(t), u(t)\right) \\ y(t) &= \psi\left(x(t), u(t)\right) \end{aligned} \tag{2.27}$$

where x is a vector of state variables, u and y vectors of inputs and outputs of the original circuit, and φ and ψ represent the nonlinear functions. Such these

nonlinear differential equations in system level are very complicated and computationally expensive. So, there is a need here for a simpler model to approximate the input/output relationship. Let n to be the order of the reduced DNN model. Let f_{ANN} represent an MLP with input neurons representing $u(t)$, $y(t)$, and their derivatives with respect to time ($d^i y/dt^i, i = 1, 2, ..., n-1$ and $d^j u/dt^j, j = 1, 2, ..., n$), and the output neurons represent $d^n y/dt^n$. Therefore, a differential DNN can be formulated as [45]

$$\begin{aligned}
&\dot{v}_1(t) = v_2(t) \\
&\vdots \\
&\dot{v}_{n-1}(t) = v_n(t) \\
&\dot{v}_n(t) = f_{ANN}\left(v_n(t), v_{n-1}(t), ..., v_1(t), u^{(n)}(t), u^{(n-1)}(t), ..., u(t)\right)
\end{aligned} \tag{2.28}$$

where $y(t) = v_1(t)$.

The DNN model (2.28) is in a standardized format for typical nonlinear circuit simulators. As an example, the left-hand side of the equation provides the charge or the capacitor part, and the right-hand side provides the current part, which are the standard representation of nonlinear components in many harmonic balance (HB) simulators. Therefore, the proposed DNN can provide dynamic current-charge parameters for general nonlinear circuits with any number of internal nodes in original circuit.

The order n represents the effective order (or the degree of nonlinearity) of the original circuit that is visible from the input-output data. Therefore, the size of the DNN reflects the internal property of the original circuit rather than external signals

and, as such, the model does not suffer from the curse of dimensionality in multi-tone simulation. By changing the number of hidden neurons, we can easily adjust the required nonlinearity degree in the DNN model. Such simple adjustments make the model creation much easier than conventional equivalent circuit based methods where manual trial and error may be needed to create/adjust the equivalent circuit. Figure 2.4 shows the structure of a DNN.

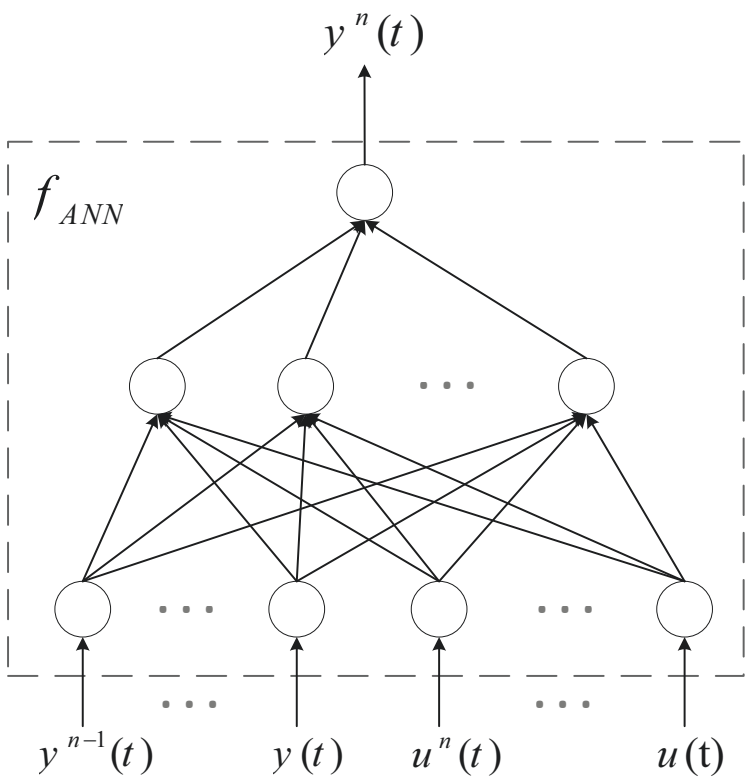


Figure 2.4: Dynamic neural network structure.

2.2 Training of ANNs

A neural network needs to be trained with corresponding training data in order to be able to represent any device/circuit behavior. Suppose $y = f(x, w)$ represents the input/output relationship of the ANN and $E(w)$ the objective function (error function) of the optimization problem (training problem). The purpose of training is to find w parameters such that the error function is minimized. As the error function is highly nonlinear with respect to w, several training algorithms have been established to accomplish this goal. In this section, some commonly used training algorithms has been reviewed.

2.2.1 Back Propagation Algorithm

Back propagation (BP) algorithm (previously mentioned in section 2.1.1) proposed by Rumelhart, Hinton, and Williams in 1986 [37], is among the most commonly used algorithms for training ANNs. In BP algorithm, weights of the neural network are updated along the negative gradient direction in the design space according to the following formula

$$\Delta w_{now} = w_{next} - w_{now} = -\eta \left. \frac{\partial E_k(w)}{\partial w} \right|_{w=w_{now}} \tag{2.29}$$

or

$$\Delta w_{now} = w_{next} - w_{now} = -\eta \left. \frac{\partial E_I(w)}{\partial w} \right|_{w=w_{now}} \tag{2.30}$$

where the learning rate, η, controls the step size of weight update. In the sample-by-sample update equation (2.29), the weights are updated after passing each training sample to the ANN. In the batch mode update equation (2.30), the weights are updated after passing all training samples to the ANN.

Since the sample-by-sample training in BP algorithm leads to a stochastic process (weight oscillation), we can keep learning rate small and add a momentum parameter to relieve this problem. By choosing a small η, we will have more epochs in the training process and the training becomes more stable. This technique (adding the momentum parameter) was introduced in [37] will modify the update equation as following

$$\Delta w_{now} = -\eta \left.\frac{\partial E_k(w)}{\partial w}\right|_{w=w_{now}} + \alpha \Delta w_{old} = -\eta \left.\frac{\partial E_k(w)}{\partial w}\right|_{w=w_{now}} + \alpha \left(w_{now} - w_{old}\right) \tag{2.31}$$

$$\Delta w_{now} = -\eta \left.\frac{\partial E_I(w)}{\partial w}\right|_{w=w_{now}} + \alpha \Delta w_{old} = -\eta \left.\frac{\partial E_I(w)}{\partial w}\right|_{w=w_{now}} + \alpha \left(w_{now} - w_{old}\right) \tag{2.32}$$

where α is the momentum factor that curbs the effect of the previous weight update direction on the current weight update, and w_{old} represents the previous value of w.

Many researchers have worked on improving the BP algorithm. In [46], two methods for increasing the performance of BP algorithm was presented, first focuses on learning rate adaptation to reduce the energy value of the gradient direction in an optimum way, and the second is derived from the conjugate gradient method

with inexact linear searches. An enhanced BP approach for learning algorithm was presented in [47] in order to reduce the learning time compared to the conventional method. In [48], an efficient method of deriving the first and second derivatives of the objective function with respect to the learning rate is presented, which does not include computation of second-order derivatives in weight space, but rather uses the gathered information from the backward and forward propagation. This method focuses on dynamic learning rate optimization of the BP algorithm using derivative information. In [49], to overcome the oscillations, a method was presented to correct the value of the weights near the bottom of a error surface ravine and a new acceleration algorithm based on that correction was introduced.

2.2.2 Gradient-Based Training Techniques

The BP algorithm explained above is relatively simple to understand and implement. However, the rate of convergence also gets slow around the ravine area. Because supervised learning of neural networks can be considered as an optimization problem, higher-order optimization methods using gradient information can be used for training in order to improve the convergence rate. Compared to the BP algorithm, these approaches have a theoretical basis and guaranteed convergence. In [50], some of the early works in this area were discussed. In [51], the first- and second-order optimization techniques for learning in feedforward neural networks was discussed. In the next parts, two most common gradient-based techniques, conjugate gradient and quasi-Newton methods, are discussed.

Conjugate Gradient Method

The conjugate gradient techniques was originally derived from quadratic minimization. By initializing the weight vector $w_{initial}$, the gradient $\frac{\partial E_I(w)}{\partial w}\Big|_{w=w_{initial}}$, and direction vector $h_{initial} = -g_{initial}$, the vector sequences of g and h are constructed recursively using conjugate gradient method as following [52]

$$g_{next} = h_{now} + \lambda_{now} H h_{now} \tag{2.33}$$

$$h_{next} = -g_{next} + \gamma_{now} h_{now} \tag{2.34}$$

$$\lambda_{now} = \frac{g_{now}^T g_{now}}{h_{now}^T H h_{now}} \tag{2.35}$$

$$\gamma_{now} = \frac{g_{next}^T g_{next}}{g_{now}^T g_{now}} \tag{2.36}$$

or,

$$\gamma_{now} = \frac{(g_{next} - g_{now})^T g_{next}}{g_{now}^T g_{now}} \tag{2.37}$$

where H is the Hessian matrix of the objective function E_I. Equation (2.36) is called the Fletcher-Reeves formula [53] and equation (2.37) the Polak-Ribiere formula [54]. To avoid the need for Hessian matrix calculation for finding the conjugate direction, another way was advanced to compute the conjugate direction [55]. First, calculate w_{next} by proceeding from w_{now}, along the direction h_{now} to the local minimum

through line minimization, and then set $g_{next} = \left.\frac{\partial E_I(w)}{\partial w}\right|_{w=w_{next}}$. This g_{next} is then used as the vector of (2.33). In this way, there is no need for computationally expensive matrix calculation. Therefore, conjugate gradient techniques are very efficient and scalable with the size of networks.

Quasi-Newton Method

In quasi-Newton method, the second-order information about the error function is used for updating the weights without the knowledge about Hessian matrix H. This method has faster convergence rate compared to conjugate gradient method because of appropriate approximation of the inverse Hessian matrix. Let A be the inverse of the Hessian matrix H. In Quasi-Newton method, the direction is calculated by modifying gradient vector g using matrix A. The weights are updated as [56]

$$w_{next} - w_{now} = -\eta A_{now} g_{now} \tag{2.38}$$

$$A_{now} = A_{old} + \Delta A_{now} \tag{2.39}$$

$$A_{now} = A_{old} + \frac{\Delta w \Delta w^T}{\Delta w^T \Delta g} - \frac{A_{old} \Delta g \Delta g^T A_{old}}{\Delta g^T A_{old} \Delta g} \tag{2.40}$$

or

$$A_{now} = A_{old} + \left(1 + \frac{\Delta g^T A_{old} \Delta g}{\Delta w^T \Delta g}\right) \frac{\Delta w \Delta w^T}{\Delta w^T \Delta g} - \frac{\Delta w \Delta g^T A_{old} + A_{old} \Delta g \Delta w^T}{\Delta w^T \Delta g} \tag{2.41}$$

where

$$\Delta w = w_{now} - w_{old} \tag{2.42}$$

$$\Delta g = g_{now} - g_{old} \tag{2.43}$$

The equation (2.40) is called the Davidon-Fletcher-Powell (DFP) formula [57] and equation (2.41) the Broyden-Fletcher-Goldfarb-Shanno (BFGS) formula [58].

Because of the huge units of required for approximation of the inverse Hessian matrix, this method will not be efficient for large networks. In limited-memory (LM) or one-step BFGS method [59], the approximation of inverse Hessian is reset to the identity matrix after every iteration and therefore eliminating the need for storage. Several approaches for parallel implementation of second-order, gradient-based MLP training algorithms was introduced in [60]. Through the approximation of inverse Hessian matrix, Quasi-Newton method has a faster convergence rate than the conjugate-gradient method.

2.3 Summary and Conclusion

In this chapter, a literature review of the ANN-based approaches for electrical and microwave modeling and design has been presented. Several types of neural network structures that are widely used in nonlinear modeling were discussed. In addition to that, several training techniques for neural networks such as back propagation algorithm, conjugate gradient method and Quasi-Newton technique have been presented.

Conventional methods for modeling the behavior of the nonlinear circuits either rely on intensive computations such as detailed transistor-level models or have the problem of limited accuracy such as equivalent-circuit-based models. The ANN-based techniques shown to have a great capability to capture both speed and accuracy advantage for modeling the nonlinear circuit even if the internal details of the circuit is not known. In the next chapters several advances over the current ANN techniques are presented for both static and dynamic transient modeling of nonlinear circuits.

Chapter 3

Parametric Modeling of Microwave Passive Components Using Sensitivity-Analysis-Based Adjoint Neural-Network Technique

This chapter presents a novel sensitivity-analysis-based adjoint neural-network (SAANN) technique to develop parametric models of microwave passive components. This technique allows robust parametric model development by learning not only the input-output behavior of the modeling problem, but also derivatives obtained from electromagnetic (EM) sensitivity analysis. A novel derivation is introduced to allow complicated high-order derivatives to be computed by a simple artificial neural-network (ANN) forward-back propagation procedure. New formulations are deduced for exact second-order sensitivity analysis of general multilayer neural-network structures with any numbers of layers and hidden neurons. Compared to the previous work on adjoint neural networks, the proposed SAANN is easier to

implement into an existing ANN structure. The proposed technique allows us to obtain accurate and parametric models with less training data. Another benefit of this technique is that the trained model can accurately predict derivatives to geometrical or material parameters, regardless of whether or not these parameters are accommodated as sensitivity variables in EM simulators. Once trained, the SAANN models provide accurate and fast prediction of EM responses and derivatives used for high-level optimization with geometrical or material parameters as design variables. Three examples including parametric modeling of coupled-line filters, cavity filters, and junctions are presented to demonstrate the validity of this technique.

3.1 Introduction

Artificial neural network (ANN) techniques have been recognized for modeling and optimization of microwave components and circuits in electromagnetic (EM)-based microwave design. [61]-[65]. Design optimization often requires repetitive adjustments of the values of geometrical or material parameters and can be very time consuming. An ANN can learn EM responses as a function of geometrical variables through an automated training process, and the trained ANN model can be subsequently implemented in high-level circuit and system designs, allowing fast simulation and optimization [55]. To improve learning and generalization in ANNs, knowledge-based neural network approaches that incorporate prior knowledge, such as analytical expressions [66], empirical models [67], [68] or equivalent circuits [69], [70], into the model structure were developed. Using these techniques, accurate models can be built with less hidden neurons and trained with less data, therefore

speeding up model development.

Recent advances in electromagnetic simulation have led to the availability of sensitivity information in addition to EM simulations, such as [71]-[74]. An algorithm for efficient estimation of S-parameter sensitivities with the time-domain transmission line modeling (TLM) method has been proposed in [75]. A time-domain algorithm for wideband adjoint variable method (AVM) sensitivity analysis for dispersive materials is presented in [76]. An adjoint sensitivity based topology optimization method for the design of patch antennas is developed in [77]. A self-adjoint sensitivity analysis based approach for enhancing the bandwidth of narrow band antennas is introduced in [78]. Also an algorithm for accelerating the space mapping optimization using adjoint sensitivities is shown in [79]. Here we propose to exploit such sensitivity information to further enhance the efficiency and accuracy of ANN models for microwave passive components. In order to train ANN models to learn EM sensitivities, we need to use ANN outputs to represent these sensitivities. Furthermore, in order to train the sensitivity-based ANN models, we need the derivatives of sensitivity outputs, therefore leading to the need of both first- and second- order derivatives in ANN. The subject of ANN derivatives has been investigated in the ANN community, and several techniques for ANN sensitivity computation have been used to train ANN models, such as [37], [80]-[82]. The most widely used method in ANN area was the back propagation method, which was one of key milestones propelling the ANN research into mainstream in the 1980s [37]. The back propagation method used a systematic mechanism to propagate ANN training error starting from the output layer and down to the input layer. Through

this process, the first-order derivatives of ANN outputs versus inputs are obtained efficiently [80]. Another interesting ANN derivative method, a generalized recursive least square method incorporating first-order derivatives for the ANN training, was developed to improve the generalization ability of ANN models while getting a compact structure [81]. Furthermore, an ANN and its extension of derivatives were applied to predict the radar cross section of a nonlinearly loaded antenna [82]. All these methods are based on first-order derivatives in ANNs. An interesting method for second-order derivative computation in ANN was presented in [83], where a rather generic neural network structures was assumed, including knowledge based neural network structure.

In this chapter, we propose a novel sensitivity-analysis-based adjoint neural network (SAANN) technique, which allows robust parametric model development by learning not only the input-output behavior of the EM modeling problem, but also the derivatives from EM sensitivity analysis. To simultaneously learn input-output behavior and the derivative information, a novel derivation is introduced to allow complicated high-order derivatives to be computed by a simple ANN backward-forward propagation procedure that can be conveniently accommodated by the existing ANN. When the model was obtained by the existing derivative data using the proposed technique, it can calculate the derivatives for the points where their derivatives were not existed. New formulations are deduced for general multilayer neural network structures with any numbers of layers and hidden neurons. Compared to the previous work [83], the proposed SAANN technique is easier and simpler to implement into an existing ANN structure. The SAANN technique,

which incorporates the derivative information into the model training process, can enhance the capability of learning and generalization of parametric models. This technique introduces a new way to reduce the amount of training data needed in the model training process while retaining model accuracy. This is beneficial because generation of training data from EM simulation or measurement is often the major expense of model development process, and thus the SAANN technique makes model development faster. Another benefit of this technique is that the trained model can be used to predict the derivative information with respect to any inputs of the model (geometrical or material variables), no matter if they are accommodated as sensitivity variables in EM simulation or not. Once trained, the SAANN models provide accurate and fast prediction of the EM responses and their corresponding derivatives used for high-level design optimization with geometrical and material parameters as design variables. The validity of the proposed approach is confirmed by three parametric modeling examples involving coupled-line filters, cavity filters and junctions.

3.2 Analysis and Incorporation of Derivative Information into Model Training Process

We propose to use the EM derivative information to train ANN models for EM problems. Let x and y represent the inputs and outputs of the original EM problems, respectively. Consider two cases of ANN learning of EM problems, ANN 1 learning only the EM input-output relationship ($x - y$ relationship, e.g., geometry versus S-parameters in the original microwave modeling problems), and ANN2 learning

not only the $x - y$ relationship, but also the dy/dx to x relationship. We illustrate the learning using 3 training samples and two testing samples as shown in Fig. 3.1. With the conventional approach, ANN1 learns the three training samples well; however the trained ANN is not accurate at testing points unless more training data are added. Our proposed approach is to train the ANN (i.e., ANN2 in the figure) to learn not only the three training samples but also the exact derivatives of dy/dx at these 3 training samples. From this figure, the training error of the typical ANN1 is small but its testing error is quite large. However, by learning training samples and their exact derivatives simultaneously, the proposed ANN2 can match well not only training samples but also testing samples.

To further investigate such accuracy advantage of the proposed sensitivity training method, we use symbol $f_0(x)$ to represent the original $x - y$ relationship of the EM problems. Suppose that in theory $f_0(x)$ has continuous derivatives of any orders. Let x_0 be a training sample. Let $f_1(x)$ be the fitting curve by the conventional ANN approach (i.e., ANN1 trained without using derivative data). Let $E_1(x_0)$ be the training error between $f_1(x)$ and $f_0(x)$ at training sample location x_0. Let $f_2(x)$ be the fitting curve by the proposed ANN (i.e., ANN2 trained with derivative data). Let $E_2(x_0)$ and $E'_2(x_0)$ represent the training errors between $f_2(x)$ and $f_0(x)$ at x_0, and between ANN derivatives $f'_2(x)$ and derivative training data $f'_0(x)$ at x_0, respectively. Based on the Taylor expansion at x_0, the models $f_0(x)$, $f_1(x)$ and $f_2(x)$ can be expanded as,

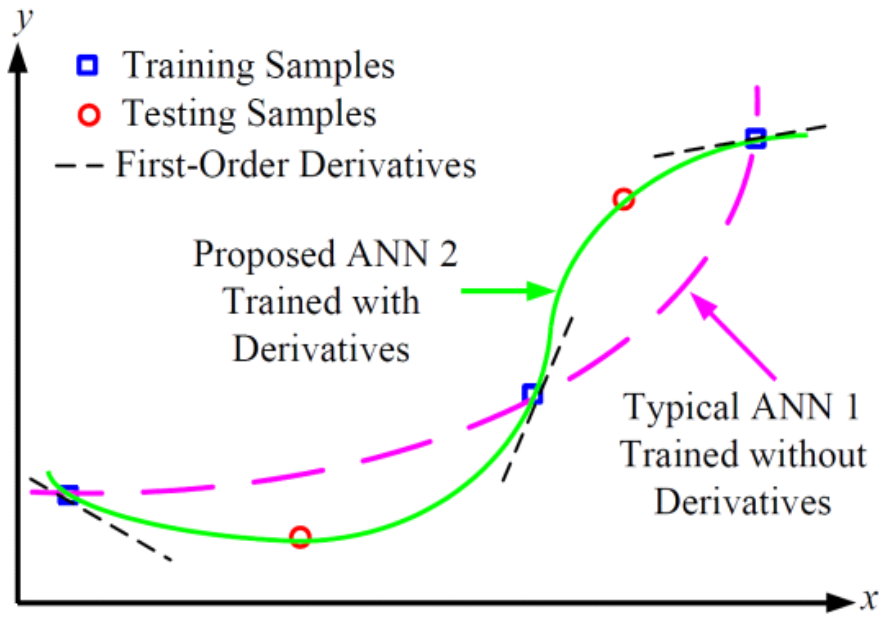


Figure 3.1: Graphical illustration of ANN learning of $x - y$ relationship with or without using dy/dx information. Trained without derivatives, the typical ANN1 can obtain a small training error but a larger testing error. Trained with derivatives, the new ANN2 can obtain a small training error and a consistent testing error.

$$
\begin{aligned}
f_0(x) &= f_0(x_0) + f_0'(x_0) \cdot \Delta x + \sum_{i=2}^{n} \frac{1}{i!} f_0^{(i)}(x_0) \cdot \Delta x^i \\
f_1(x) &= f_0(x_0) + E_1(x_0) + f_1'(x_0) \cdot \Delta x + \sum_{i=2}^{n} \frac{1}{i!} f_1^{(i)}(x_0) \cdot \Delta x^i \\
f_2(x) &= f_0(x_0) + E_2(x_0) + f_0'(x_0) \cdot \Delta x + E_2'(x_0) \cdot \Delta x + \sum_{i=2}^{n} \frac{1}{i!} f_2^{(i)}(x_0) \cdot \Delta x^i
\end{aligned}
\tag{3.1}
$$

In the ideal case, when the proposed and conventional ANNs are both trained very well, the training errors $E_1(x_0)$, $E_2(x_0)$ and $E_2'(x_0)$, will be all equal to zeros. Assuming higher order parts of the equations are negligible because of small Δx^i, in such ideal case, the testing errors of proposed and conventional ANNs at testing sample $x = x_0 + \Delta x$ are,

$$E_1(x_0{+}\Delta x) = |f_1(x) - f_0(x)| = \left| [f_1'(x_0) - f_0'(x_0)] \cdot \Delta x + \sum_{i=2}^{n} \frac{1}{i!} \left[f_1^{(i)}(x_0) - f_0^{(i)}(x_0) \right] \cdot \Delta x^i \right|$$

$$E_2(x_0 + \Delta x) = |f_2(x) - f_0(x)| = \left| \sum_{i=2}^{n} \frac{1}{i!} \left[f_2^{(i)}(x_0) - f_0^{(i)}(x_0) \right] \cdot \Delta x^i \right|$$

Clearly,

$$\lim_{\Delta x \to 0} \frac{E_2 - E_1}{\Delta x} = -\left| f_1'(x_0) - f_0'(x_0) \right| \leq 0 \tag{3.2}$$

Therefore, the testing error E_2 of the proposed ANN2 is absolutely lower than the testing error E_1 of the conventional ANN1 if the testing sample $x_0 + \Delta x$ is not far from the training sample x_0.

3.3 Proposed Sensitivity-Analysis-Based Adjoint Neural Network Technique

3.3.1 Structure of the Proposed SAANN Model

Let x be a vector representing the inputs of the original neural network such as frequency, geometrical and material parameters of microwave passive components.

Let y be a vector representing the outputs of the original neural network such as real and imaginary parts of S-parameters (scattering parameters are the elements of a scattering matrix describing the electrical behavior of linear electrical systems). Let w represent the synaptic weights of original neural network. The adjoint neural network is a "companion" neural network sharing the same set of internal neuron-connection parameters as that in original neural network, but with modified neuron activation functions such that the adjoint neural network provides first-order derivative information dy/dx. The detailed explanation of the adjoint neural network concept is in [83].

The structure of the sensitivity-analysis-based adjoint neural network and its training is shown in Fig. 3.2. The SAANN model consists of two parts: the original neural network and the adjoint neural network. The inputs x of the SAANN model contain frequency, geometrical and material parameters, which are the same as those of the original neural network. The outputs of the SAANN model contain the outputs y of the original neural network in addition to the derivatives dy/dx, which are the outputs of the adjoint neural network. Let d and d' be vectors representing the outputs of EM simulations (e.g, S parameters) and the derivatives of S-parameters with respect to geometrical or material variables from EM sensitivity analysis, respectively. The object of the SAANN training is to adjust the internal weights w such that for all training samples the error between y and the training data d and dy/dx and d' are minimized. Although the whole training process involves both the original and adjoint neural networks, the final parametric model can be fairly simple only containing the original neural network as shown in Fig. 3.2. Let

the total training error be defined as

$$E_T = E_o + E_a = \frac{1}{2}A\sum_{q\in Q}(y_q - d_q)^2 + \frac{1}{2}\sum_{p\in P, q\in Q} B_{q,p}\left(\frac{\partial y_q}{\partial x_p} - d'_{q,p}\right)^2 \tag{3.3}$$

where E_o and E_a represent the training error from original and adjoint neural network models, respectively, x_p and y_q denote the pth input in x and the qth output in y, respectively. P and Q represent index sets of inputs and outputs, respectively. $d'_{q,p}$ represent the training data for the derivative of the qth output with respect to the pth input. A and $B_{q,p}$ are the weighting factors for different terms in the error function (3.3), e.g., A representing the inverse of the minimum to maximum range of training data d_q, for $q \in Q$, and $B_{q,p}$ representing the inverse of the minimum to maximum range of training data $d'_{q,p}$, for $p \in P$,$q \in Q$.

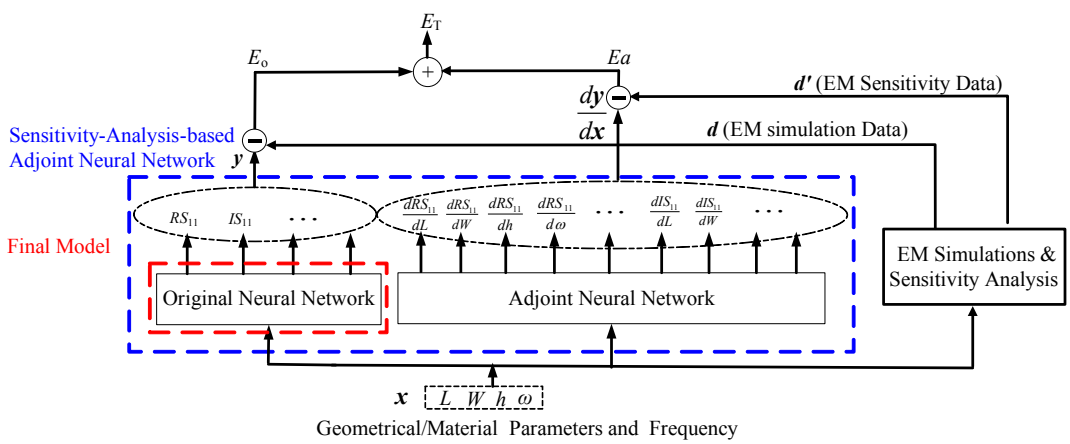


Figure 3.2: Structure of the proposed SAANN model. It consists of two parts: original neural network and adjoint neural network, where L, W, h, and ω represent geometrical parameters such as length, width, thickness of substrates and frequency, respectively.

3.3.2 Second-order derivatives for Training the SAANN Model

During the traditional ANN training process, only first-order derivatives are required to guide the gradient based training process, and such first-order derivatives can be computed through the back propagation method [55]. In order to train the original and adjoint neural network efficiently and simultaneously, the second-order derivatives with respect to ANN internal weights w should also be found.

The structure of the original neural network, as shown in Fig. 3.3, contains multi-layers with the sigmoid function as the activation function in each hidden neuron. Different from our previous work [83] where the second-order derivatives were calculated through a special computation process different from the original ANN, here a novel derivation is introduced to allow complicated second-order derivatives to be computed by a simpler ANN forward-backward propagation procedure, which can be conveniently accommodated by the existing ANN computational mechanism.

The proposed forward-backward propagation method is a combination of the standard back propagation procedure and a new procedure that maximally utilizes the ANN feedforward infrastructure already existing in typical ANN computations. The outputs of the ith hidden neuron in l^{th} layer of a standard multilayer perceptron (MLP) neural network are defined as [55]

$$z_i^l = \begin{cases} \gamma_i^l \quad for\, i = 1, 2, \ldots, N_l, l = L \\ \sigma\left(\gamma_i^l\right) \quad for\, i = 1, 2, \ldots, N_l, l = 2, \ldots, L-1 \\ x_i \quad for\, l = 1 \end{cases} \tag{3.4}$$

where

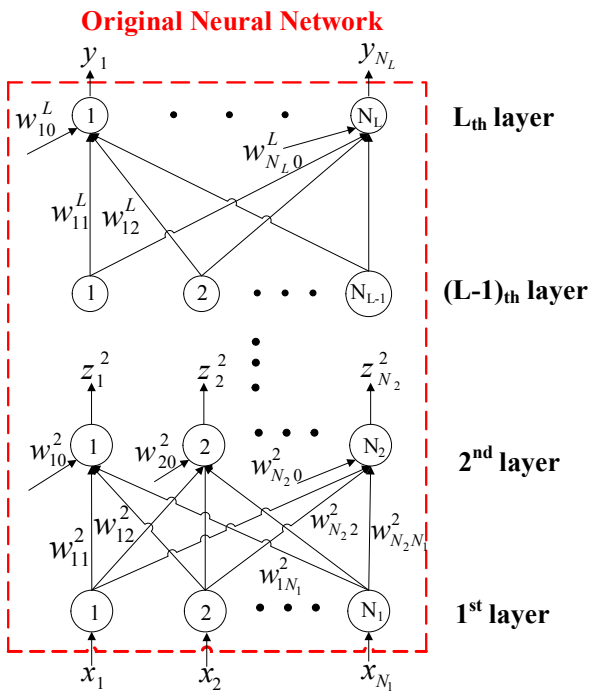


Figure 3.3: Structure of the original neural network.

$$\gamma_i^l = \sum_{k=0}^{N_{l-1}} w_{ik}^l . z_k^{l-1} \, (i = 1, 2, \ldots, N_l l = 2, 3, \ldots, L) \tag{3.5}$$

and w_{ik}^l is the weight between i^{th} hidden neuron of the lth layer and k^{th} hidden neuron of the $(l-1)^{th}$ layer, y_i is the ith output of the original neural network, $\sigma(\gamma)$ is the sigmoid function, N_l is the total number of hidden neurons in the lth layer, and L is the total number of layers. Note that for simplicity of the bias calculation, the first neuron in each layer is supposed to be 1, i.e., $z_0^l = 1 (l = 1, 2, ..., L)$.

To calculate the second-order derivatives efficiently, we define new variables of

α_{qi}^{l} and β_{ip}^{l} as,

$$\alpha_{qi}^{l} = \frac{\partial y_q}{\partial \gamma_i^l} \tag{3.6}$$

$$\beta_{ip}^{l} = \frac{\partial z_i^l}{\partial x_p} \tag{3.7}$$

where $l = 2, ..., L$; $q = 1, ..., N_L$; $i = 1, ..., N_l$; $p = 1, ..., N_1$

According to the definition of α_{qi}^{l}, for the last layer, i.e. $l = L$, α_{qi}^{L} is initialized as

$$\alpha_{qi}^{L} = \begin{cases} 1, i = q \\ 0, i \neq q \end{cases} \quad i = 1, 2, \ldots, N_L \quad q = 1, 2, \ldots, N_L \tag{3.8}$$

Then, α_{qi}^{l} can be recursively calculated using the back propagation procedure,

$$\alpha_{qi}^{l} = \sum_{k=1}^{N_{l+1}} \frac{\partial y_q}{\partial \gamma_k^{l+1}} . \frac{\partial \gamma_k^{l+1}}{\partial \gamma_i^l} = z_i^l . \left(1 - z_i^l\right) \sum_{k=1}^{N_{l+1}} \alpha_{qk}^{l+1} . w_{ki}^{l+1} \tag{3.9}$$

for $i = 1, ..., N_l$; $l = L - 1, ..., 2$; $q = 1, ..., N_L$. This process is further illustrated in Fig. 3.4.

Now the adjoint neural network can be built using the αs as shown in Fig. 3.5. The outputs of the adjoint neural network, i.e, the derivative of the outputs y of the original neural network to inputs x can be calculated as,

$$\frac{\partial y_q}{\partial x_p} = \sum_{i=1}^{N_2} \frac{\partial y_q}{\partial \gamma_i^2} . \frac{\partial \gamma_i^2}{\partial x_p} = \sum_{i=1}^{N_2} \alpha_{qi}^2 . w_{ip}^2 \tag{3.10}$$

where $p = 1, ..., N_1$, $q = 1, ..., N_L$, γ_i^2 is γ_j^l in Equation (3.5) at the second layer, and w_{ip}^2 is the weight between i^{th} hidden neuron of the second layer and p^{th} hidden

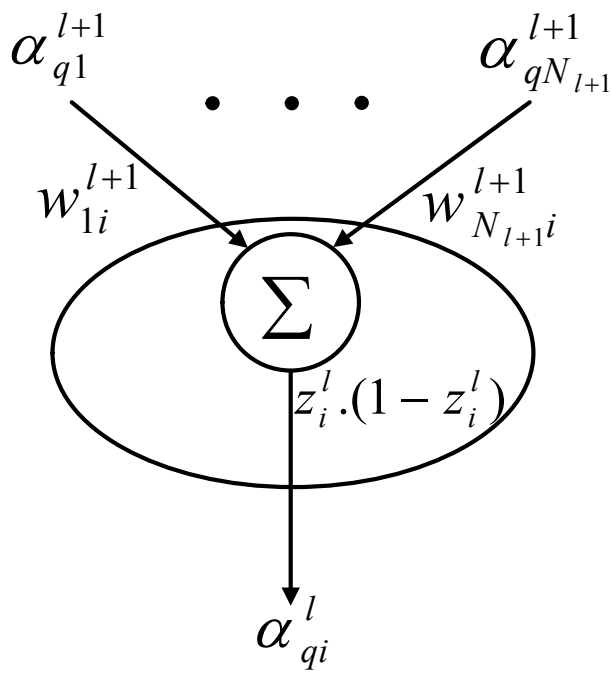


Figure 3.4: Calculation of the proposed parameter α using the back propagation procedure available from the standard ANN procedure.

neuron of the first layer. This process is the same as the standard back propagation procedure [55] except that the starting error vector for back-propagation is a binary vector defined by (3.8) for a fixed q. In this way, the proposed parameters α are obtained with the minimum change to the standard ANN implementation. In this chapter, an Adjoint Neural Network is defined to represent the computation of the first order derivatives in the original neural network. The adjoint neural network is illustrated in Fig. 3.5 (only derivatives of one of the outputs of the ANN to all the inputs are shown in this figure). The output of the adjoint neural network

represents the derivative of the original neural network output with respect to the original neural network inputs. As seen in Fig. 3.5, the adjoint neural network is the reverse of the original neural network so that the number of inputs of the adjoint neural network is the number of outputs of the original neural network.

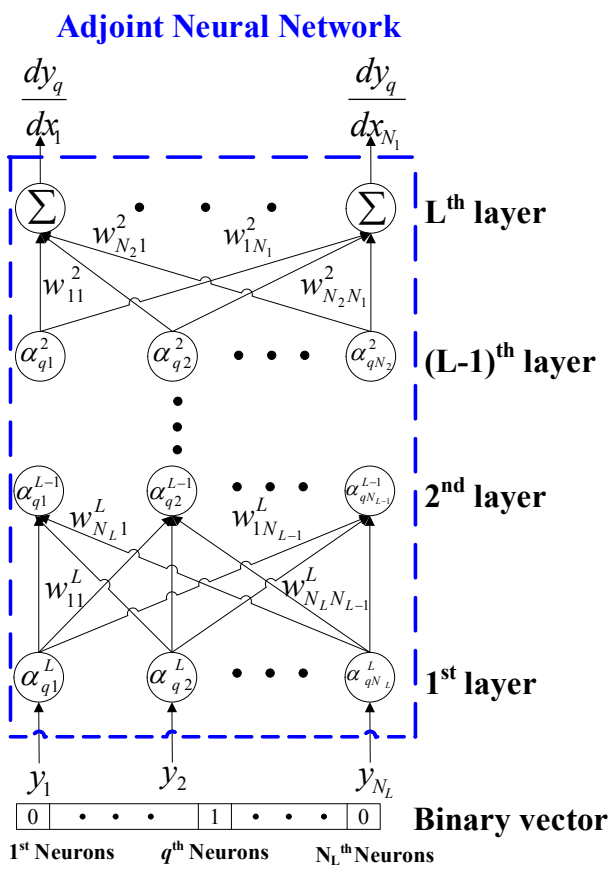


Figure 3.5: The structure of the adjoint neural network (the outputs demonstrated here include only the derivatives of one of the outputs of the ANN to all of the inputs) using back propagation calculation of α_{qi}^{l} for each layer. As it can be seen the last layer computations contain only summation without an extra multiplication.

Next, we derive a simple method to compute β_{ip}^{l} for each layer by maximally utilizing the ANN feedforward infrastructure already existing in typical ANN computations. For each given index p, we formulate a systematic recursive procedure starting at the input layer. For the first layer, β_{ip}^{1} is initialized as

$$\beta_{ip}^{1} = \begin{cases} 1 & i = p \\ 0 & i \neq p \end{cases} \tag{3.11}$$

for $i = 1, ..., N_l$; $p = 1, ..., N_1$.

The next step using feedforward procedure is to compute β_{ip}^{l} for the upper layers. According to definitions of z_i^l in Equation (3.4) and γ_i^l in Equation (3.5),

$$\begin{aligned} \beta_{ip}^{l} &= \frac{\partial z_i^l}{\partial \gamma_i^l} . \frac{\partial \gamma_i^l}{\partial x_p} = z_i^l . \left(1 - z_i^l\right) . \frac{\partial \left(\sum_{k=0}^{N_{l-1}} w_{ik}^l . z_k^{l-1}\right)}{\partial x_p} \\ &= z_i^l . \left(1 - z_i^l\right) . \sum_{k=1}^{N_{l-1}} w_{ik}^l . \beta_{kp}^{l-1} \qquad \begin{array}{c} i = 1, \ldots, N_l \\ l = 2, \ldots, L-1 \end{array} \end{aligned} \tag{3.12}$$

According to the definitions of z_i^l in Equation (3.4), z_i^l for the last layer is computed differently from other layers. Thus, the last step after calculating β_{ip}^{l} for all layers lower than L, is to compute β_{ip}^{L} as

$$\beta_{ip}^{L} = \frac{\partial z_i^L}{\partial x_p} = \frac{\partial \gamma_i^L}{\partial x_p} = \sum_{k=0}^{N_{l-1}} w_{ik}^L . \beta_{kp}^{L-1} \quad i = 1, \ldots, N_L \tag{3.13}$$

Fig. 3.6 shows the inside of a typical β_{ip}^{l} block. It includes a multiplication after

the summation. From this figure, we can see this block is very similar to a node in the original neural network structure except that the activation function in each neuron is a multiplication of $z_i^l(1 - z_i^l)$ instead of the sigmoid function.

Similar to the calculation of the first derivative information, there is another binary vector in the process of the calculation of but with the length of N_1 so that at the same time just one of the elements is 1 and it determines which x_p is selected for feedforward computation. Fig. 3.7 shows one step standard feedforward in the forward propagation method for calculating β.

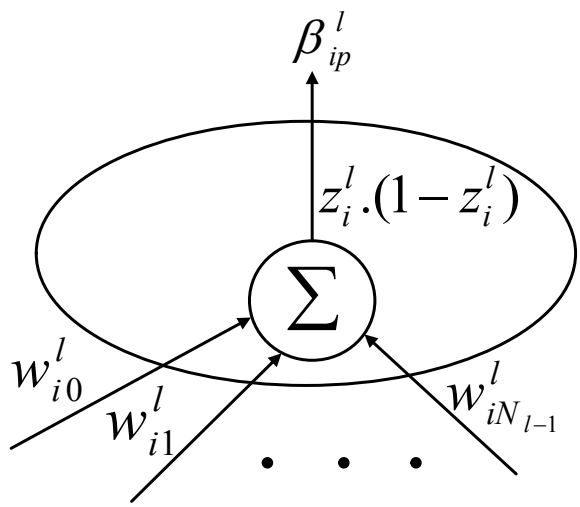


Figure 3.6: Block diagram of β_{ip}^l. As shown in this figure, this block is very similar to a node in the original neural network structure except that the activation function in each neuron is a multiplication of $z_i^l(1 - z_i^l)$ instead of the sigmoid function.

Based on the calculation of α and β, the second-order derivatives can be obtained. We define

$$\theta^l_{qip} = \frac{\partial^2 y_q}{\partial \gamma^l_i \partial x_p} \tag{3.14}$$

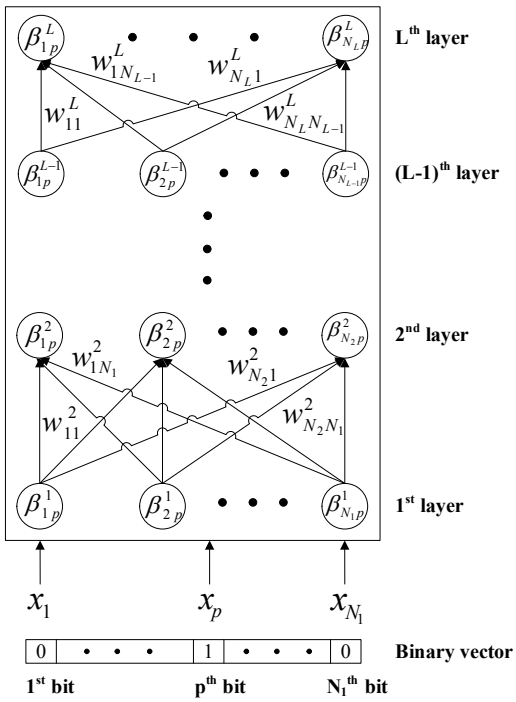


Figure 3.7: One sample feedforward step in forward propagation method for the calculation of β for x_p. From this figure, we can see the calculation of β can be done within the original neural network structure except that the activation function is a multiplication of $z^l_i(1 - z^l_i)$ instead of the sigmoid function.

Firstly, for the output layer, i.e, the layer $l = L$, θ^L_{qip} needs to be initialized. According to the definition in Equation (3.14), the first-order derivative of y_q to γ^L_i in the ith neuron in output layer can be obtained as

$$\frac{\partial y_q}{\partial \gamma_i^L} = \begin{cases} 1 & q = i \\ 0 & q \neq i \end{cases}$$

Since the above derivative is a constant value, its second-order derivative to input x_p is zero, i.e.,

$$\frac{\partial}{\partial x_p}\left(\frac{\partial^2 y_q}{\partial \gamma_i^L}\right) = 0$$

Thus for the output layer, θ_{qip}^L is initialized as

$$\theta_{qip}^L = \frac{\partial^2 y_q}{\partial \gamma_i^L \partial x_p} = 0 \tag{3.15}$$

for $q = 1, ..., N_L$, $i = 1, ..., N_L$, and $p = 1, ..., N_1$. It indicates that for the output layer, the second-order derivatives of y_q to γ_i^L in the ith neuron and input xp are fixed to zeros.

According to the definition of θ_{qip}^l in Equation (3.14) for layers below the output layer, i.e., layer $l \neq L$,

$$\theta_{qip}^l = \frac{\partial\left(\frac{\partial y_q}{\partial \gamma_i^l}\right)}{\partial x_p} = \frac{\partial\left(\alpha_{qi}^l\right)}{\partial x_p} \tag{3.16}$$

Utilizing Equation (3.9), Equation (3.16) now becomes,

$$\theta_{qip}^l = \sum_{k=1}^{N_{l+1}} \left(\frac{\partial \alpha_{qk}^{l+1}}{\partial x_p}.w_{kq}^{l+1}.z_i^l.\left(1 - z_i^l\right) + \frac{\partial\left(z_i^l.\left(1 - z_i^l\right)\right)}{\partial x_p}.\alpha_{qk}^{l+1}.w_{kq}^{l+1} \right) \tag{3.17}$$

where utilizing the definition of β_{ip}^l in Equation (3.7),

$$\frac{\partial\left(z_i^l.\left(1-z_i^l\right)\right)}{\partial x_p}=\left(1-2.z_i^l\right).\frac{\partial z_i^l}{\partial x_p}=\left(1-2.z_i^l\right).\beta_{ip}^l$$

Therefore, θ_{qip}^l in Equation (3.17) can be calculated recursively as

$$\theta_{qip}^l=z_i^l.\left(1-z_i^l\right).\sum_{k=1}^{N_{l+1}}\theta_{qkp}^{l+1}.w_{kq}^{l+1}+\left(1-2.z_i^l\right).\beta_{ip}^l.\sum_{k=1}^{N_{l+1}}\alpha_{qk}^{l+1}.w_{kq}^{l+1} \tag{3.18}$$

for $l=L-1,...,2$; $q=1,...,N_L$; $i=1,...,N_l$; $p=1,...,N_1$

Fig. 3.8 shows the calculation of θ_{qip}^l based on θ_{qip}^{l+1} and α_{qi}^{l+1} in the upper layer using simple back propagation procedure. From this figure, we can see the calculation of θ_{qip}^l is quite similar to twice the standard back propagation calculation of the first-order derivatives in (3.9) in addition to two multiplications.

Now, the second-order derivatives of the outputs of the original neural network model, e.g., the derivatives of S-parameters to geometrical variables x, to ANN internal weights w can be computed as,

$$\frac{\partial^2 y_q}{\partial w_{ij}^l.\partial x_p}=\frac{\partial\left(\frac{\partial y_q}{\partial \gamma_i^l}.\frac{\partial \gamma_i^l}{\partial w_{ij}^l}\right)}{\partial x_p} \tag{3.19}$$

According to the Equation (3.5),

$$\frac{\partial \gamma_i^l}{\partial w_{ij}^l}=z_j^{l-1}$$

Equation (3.19) now becomes,

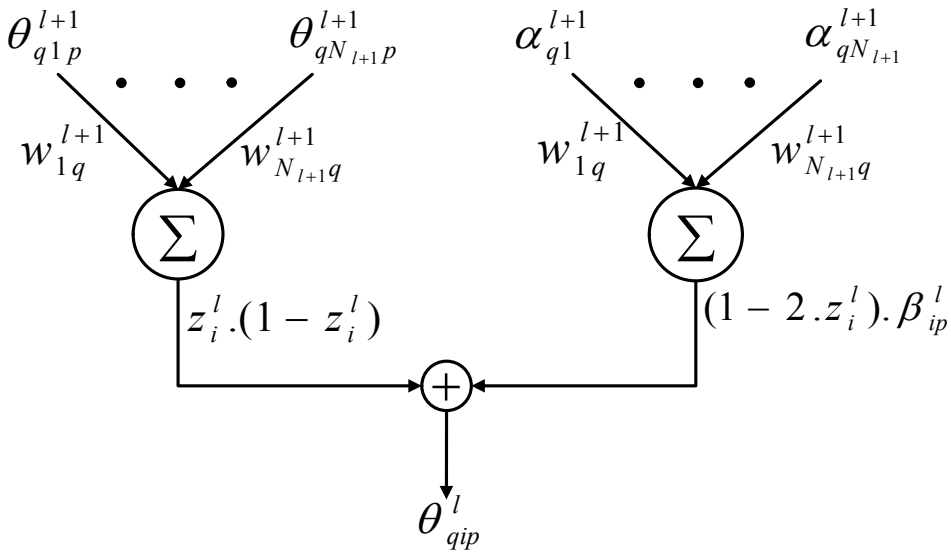


Figure 3.8: Calculation of θ^l_{qip} using back propagation procedure. As shown in this figure, the calculation of θ^l_{qip} is very similar to the calculation of the first-order derivative information in addition to some extra multiplication factors.

$$\frac{\partial^2 y_q}{\partial w^l_{ij}.\partial x_p} = \frac{\partial^2 y_q}{\partial \gamma^l_i \partial x_p}.\frac{\partial \gamma^l_i}{\partial w^l_{ij}} + \frac{\partial\left(\frac{\partial \gamma^l_i}{\partial w^l_{ij}}\right)}{\partial x_p}.\frac{\partial y_q}{\partial \gamma^l_i} \tag{3.20}$$
$$= \frac{\partial^2 y_q}{\partial \gamma^l_i \partial x_p}.z^{l-1}_j + \frac{\partial z^{l-1}_j}{\partial x_p}.\frac{\partial y_q}{\partial \gamma^l_i} = \theta^l_{qip}.z^{l-1}_j + \beta^{l-1}_{jp}.\alpha^l_{qi}$$

For $l = 2, ..., L$; $q = 1, ..., N_L$; $p = 1, ..., N_1$; $i = 1, ..., N_l$; $j = 1, ..., N_{l-1}$.

As shown in this Equation (3.20), once α^l_{qi}, β^{l-1}_{ip}, and θ^l_{qip} are computed, the second-order derivatives of the outputs y to ANN internal weights w are readily obtained. Fig. 3.9 is a block diagram demonstrating the process of calculating of the second-order derivatives for the proposed SAANN model. To obtain the

second-order derivatives, firstly β^l_{ip} has to be initialized following (3.11) with $l2$ and calculated recursively following (3.13) using the forward propagation procedure with increasing l until lL. Then, α^l_{qi} is initialized following (3.8) with lL, and calculated recursively following (3.9) using the back propagation procedure with decreasing l until $l2$. Note that the calculation of α and β can be done in parallel. Next, θ^l_{qip} is initialized following (3.15) with lL, and calculated recursively following (3.18) using computed β^l_{ip} and α^l_{qi} and the back propagation procedure with decreasing l until $l2$. Finally, the computed α, β and θ are used to calculate the second-order derivatives following (3.20).

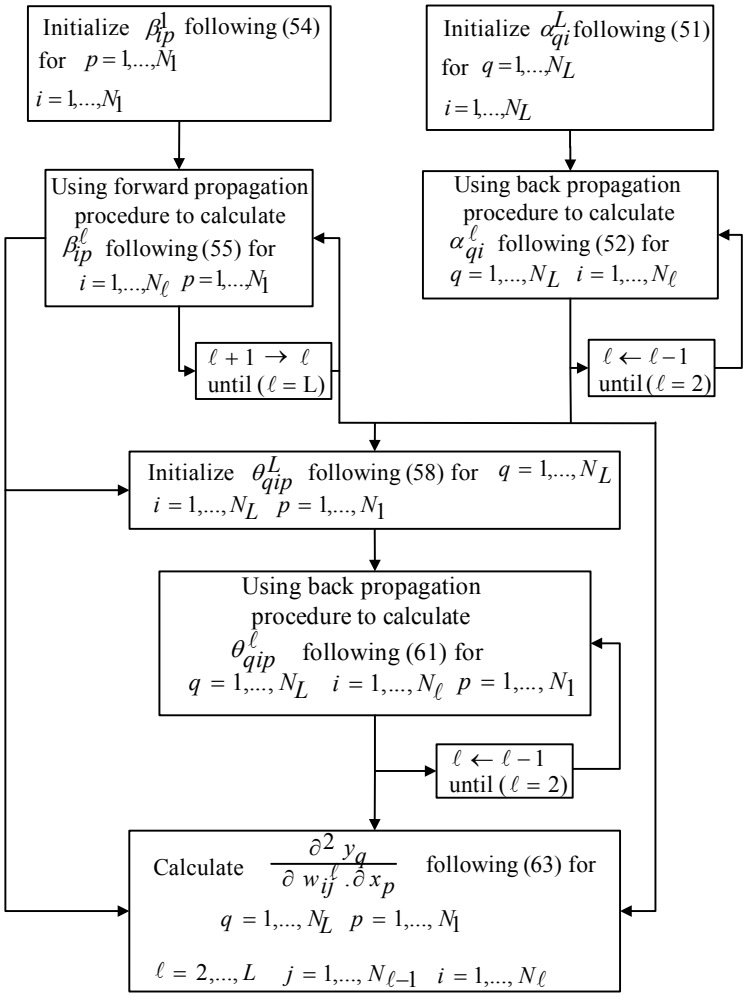


Figure 3.9: Calculation of the second-order derivatives of the proposed SAANN parametric model.

3.4 Application Examples

3.4.1 Parametric Modeling of a Coupled-Line Filter

In this example, we illustrate the use of the proposed SAANN technique to develop a parametric model for a family of coupled-line filters as shown in Fig. 3.10, where S_1 and S_2 are the spacing between lines, and D_1, D_2 and D_3 are the offset distances from the ends of each coupled lines to the corresponding fringes, respectively.

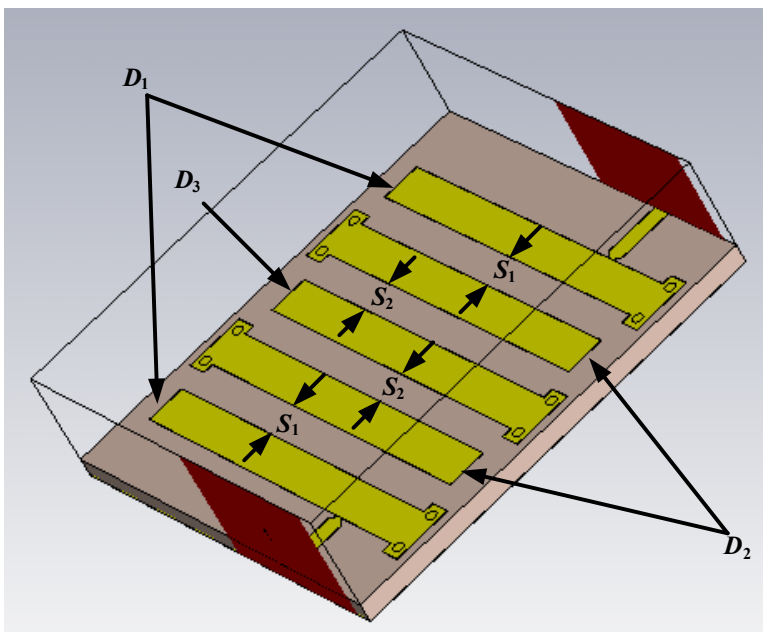


Figure 3.10: Structure of a coupled-line filter and geometrical parameters used for generating training data for parametric modeling example.

The structure of the SAANN model for the coupled-line filter example is shown in Fig. 3.11. This parametric model has six inputs i.e., $x = [S_1\, S_2\, D_1\, D_2\, D_3\, \omega]^T$,

which include five geometrical variables S_1, S_2, D_1, D_2, and D_3 defined in Fig. 3.10 and frequency ω. A 3D EM simulator, i.e., CST Microwave Studio(R) [71], is used to generate S-parameters and sensitivity information. In the implementation of sensitivity analysis in the EM simulator, the variables D_1, D_2, and D_3 are set as the sensitivity geometrical variables and the variables S_1 and S_2 are variables without sensitivity information. This SAANN model combining the original and adjoint neural networks used for training has 28 outputs, i.e., $[RS_{11}\ IS_{11}\ RS_{12}\ IS_{12}\ \frac{dRS_{11}}{dS_1}\ \frac{dRS_{11}}{dS_2}\ \frac{dRS_{11}}{dD_1}\ \frac{dRS_{11}}{dD_2}\ \frac{dRS_{11}}{dD_3}\ \frac{dRS_{11}}{d\omega}\ \frac{dIS_{11}}{dS_1}\ \frac{dIS_{11}}{dS_2}\ \frac{dIS_{11}}{dD_1}\ \frac{dIS_{11}}{dD_2}\ \frac{dIS_{11}}{dD_3}\ \frac{dIS_{11}}{d\omega}\ \ldots\ \frac{dIS_{12}}{d\omega}]^T$, which are real and imaginary parts of S_{11} and S_{12}, the derivatives of real and imaginary parts of S_{11} and S_{12} with respect to six input variables (including frequency). The sensitivity analysis in EM simulator is performed to obtain the derivatives of real and imaginary parts of S_{11} and S_{12} to three sensitivity variables D_1, D_2, and D_3. Since the other variables are not available from EM simulation (i.e., S_1, S_2 and ω are non sensitivity-variables in CST EM simulation), the corresponding outputs from SAANN model are left as free variables in the model training process. This is achieved by setting the training weights for $[\frac{dRS_{11}}{dS_1}\ \frac{dRS_{11}}{dS_2}\ \frac{dRS_{11}}{d\omega}\ \frac{dIS_{11}}{dS_1}\ \frac{dIS_{11}}{dS_2}\ \frac{dIS_{11}}{d\omega}\ \frac{dRS_{12}}{dS_1}\ \frac{dRS_{12}}{dS_2}\ \frac{dRS_{12}}{d\omega}\ \frac{dIS_{12}}{dS_1}\ \frac{dIS_{12}}{dS_2}\ \frac{dIS_{12}}{d\omega}]^T$ as zero in our training program [84].The frequency range is from 2 GHz to 2.9 GHz with a step size of 2.7 MHz. In order to show the merits of the SAANN technique which can enhance the capability of learning and generalization of the overall models with less training data, the data range of training data and testing data is defined in Table 3.1. Partial orthogonal design of experiments method [85], is used to determine the size of training and testing data. Although the whole training process involved original neural network and adjoint

neural network, the final parametric model is simple, only containing the original neural network.

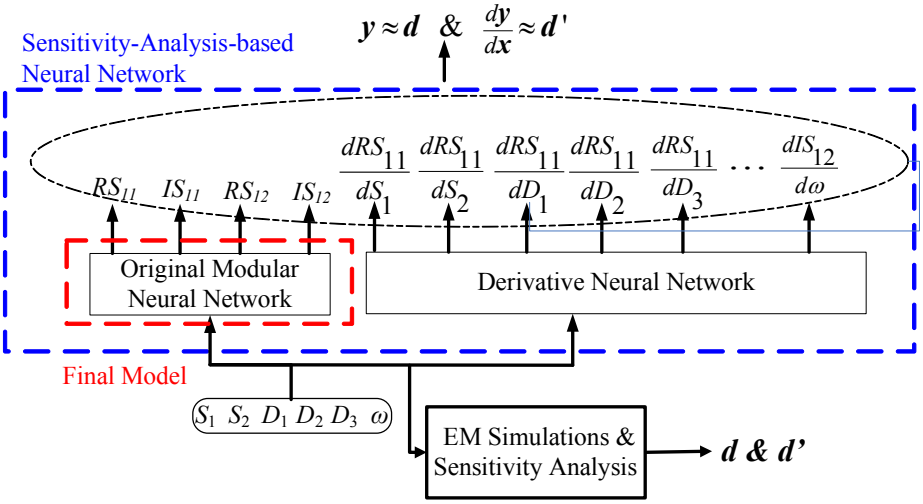


Figure 3.11: Structure of the parametric SAANN model for coupled-line filters.

Fig. 3.12 depicts the outputs of the proposed SAANN model for three different geometries #1, #2, and #3, and its comparison with EM data and conventional ANN model trained with training data of different sizes.

The geometrical variables for three coupled-line filters are as follows (negative values are based on the offset from the initial points provided by CST):

Geometry 1: S_1= 39.5 mm, S_2= 37.5 mm, D_1= 4.1 mm, D_2= -2.5 mm, D_3= -2.3 mm,

Geometry 2: S_1= 40.5 mm, S_2=38.5 mm, D_1= 7.5 mm, D_2 = -1.1 mm, D_3 = -1.1 mm,

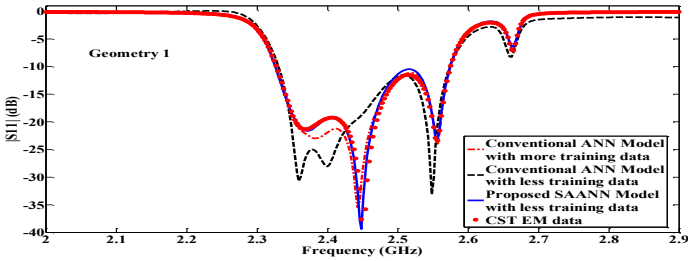
Geometry 3: S_1= 36.5 mm, S_2= 38.5 mm, D_1= 6.5 mm, D_2 = -3.1 mm, D_3 = -0.5 mm.

Table 3.1: Definition of Training and Testing Data for The Coupled-Line Filter Example

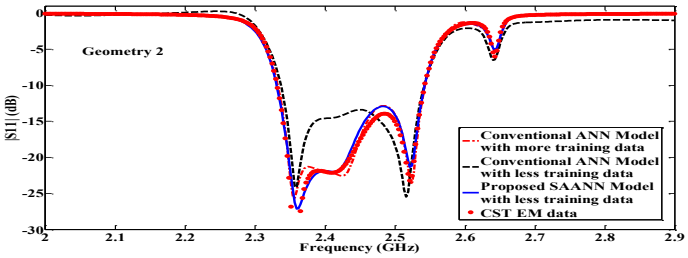
Parameters		Training data			Testing data		
		Min	Max	Step	Min	Max	Step
S_1(mm)		36	44	1	36.5	43.5	1
S_2(mm)		36	44	1	36.5	43.5	1
Sensitivity Variables	D_1 (mm)	4	8	0.2	4.1	7.9	0.2
	D_2 (mm)	-4.6	-0.6	0.2	-4.5	-0.7	0.2
	D_3 (mm)	-4.4	-0.4	0.2	-4.3	-0.5	0.2

Table 3.2: Training and Testing Results for Coupled-Line Filter Example

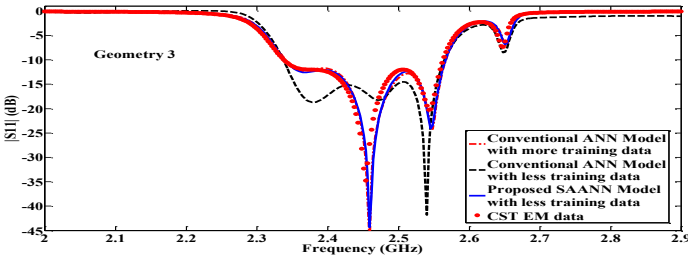
Model Type	Original Neural Network Structure	Average Training Error	Average Testing Error
Conventional ANN Model using 120 sets of training data	6-40-4	0.897%	0.989%
Conventional ANN Model using 40 sets of training data	6-35-4	1.073%	4.357%
Proposed SAANN Model using 40 sets of training data	6-35-4	0.871%	0.946%



(a)



(b)



(c)

Figure 3.12: Comparison of the magnitude in dB of S_{11} of the SAANN model trained with less data (40 sets of data), CST EM data, conventional ANN model trained with less data (40 sets of data), and conventional ANN model trained with more data (120 sets of data) for three different filter geometries. These 3 different geometries are from test data and have never been used in training. As shown in this figure, using the SAANN model, we can use less training data to achieve good model accuracy than that needed for conventional ANN model.

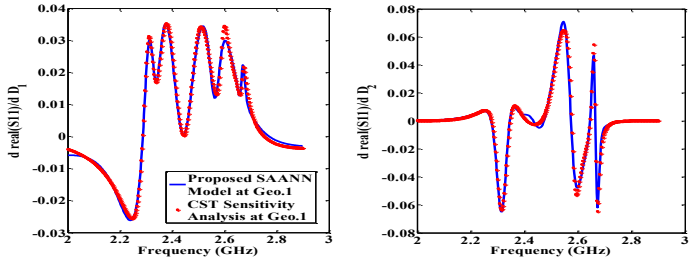
As shown in Fig. 3.12, broadband accuracy of the proposed SAANN model is confirmed by its good agreement with EM data in terms of S_{11} even though these geometries are never used in the training process.

As shown in table 3.2, SAANN trained with few data can achieve similar accuracy as conventional ANN trained with much more data. In this way, the development time for the proposed SAANN model is much shorter than that of conventional ANN. All simulations in this chapter are done on the same computer with Intel core 2 Quad CPU@2.4 GHz and 4 GB memory. The obtained ANN model achieves almost the same solutions as CST EM simulations using much less time. The SAANN model development cost for this coupled-line filter example, including training data generation time (40 sets of training geometries) and model training time, is about 5.46 hours and for conventional ANN model development (120 sets of training geometries) is about 15.5 hours. Note that the training is a one-time investment, and the benefit of using the model accumulates when the model is used over and over again.

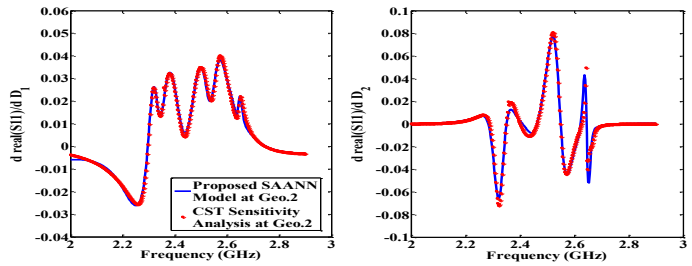
Here, we show another benefit of this proposed SAANN technique that the trained model can accurately predict the derivative information with respect to geometrical variables. As shown in Fig. 3.13, we provide the comparison of the derivative information of the real part of S_{11} with respect to sensitivity variables D_1, D_2, and D_3 by the proposed SAANN parametric model and CST sensitivity analysis at geometries #1, #2, and #3, respectively. This figure confirms that the proposed SAANN model can approximate the derivative information well, even though the geometry values have never been used in training.

In Fig. 3.14, we utilize the sensitivity ability of SAANN to predict the derivative information of the real part of S_{11} with respect to non-sensitivity variables S_1 and S_2, by the SAANN parametric model and perturbation sensitivity at geometries #1, #2, and #3, respectively. This figure demonstrates that the SAANN parametric model can be used to accurately predict the derivative information with respect to geometrical variables, which can even be non-sensitivity variables in EM simulation.

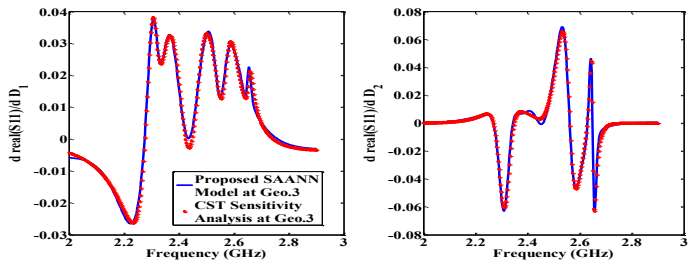
As an example to demonstrate the validity of the proposed second-order derivatives calculation in the SAANN technique, Fig. 3.15 compares the second-order derivatives of the real part of S_{11} to variables D_1 or D_2 and ANN weights w_{11}^2 and w_{11}^3 at geometries #1 by the SAANN parametric model versus that from perturbation as a continuous function in frequency sub-spaces before and after training, respectively. The good agreement in those figures verifies our proposed formulas (3.6)-(3.20) for the second-order derivatives calculation in the SAANN technique.



(a)

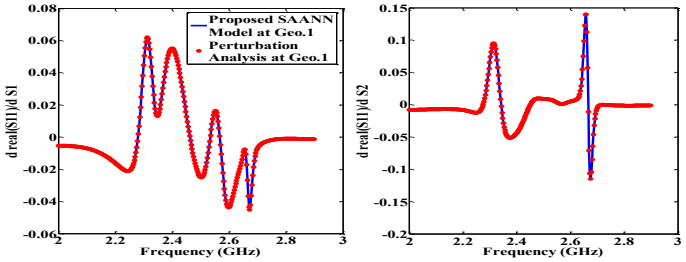


(b)

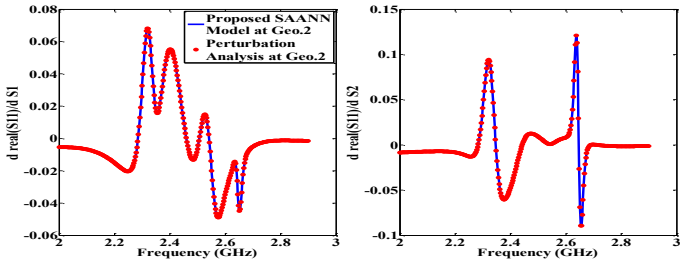


(c)

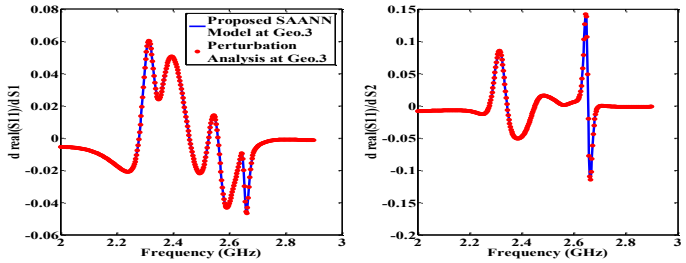
Figure 3.13: Comparison of the derivative information of the real part of S_{11} to sensitivity variables D_1, D_2, and D_3 by the proposed SAANN model and CST sensitivity analysis for $\frac{dRS_{11}}{dD_1}$, and $\frac{dRS_{11}}{dD_2}$ at (a) geometries #1, (b) geometries #2 , (c) geometries #3 for the coupled-line filter example. As shown in this figure, the proposed SAANN model can accurately predict the derivative information, which are much closer to those obtained from CST sensitivity analysis, even though such geometries are never used in the training process.



(a)

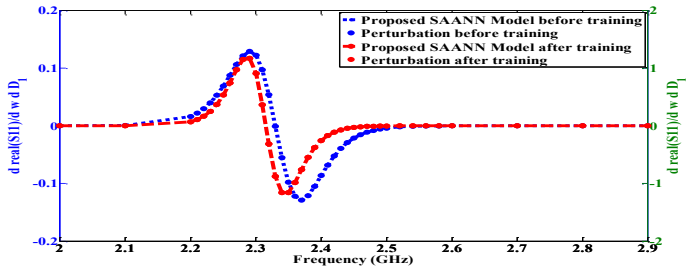


(b)

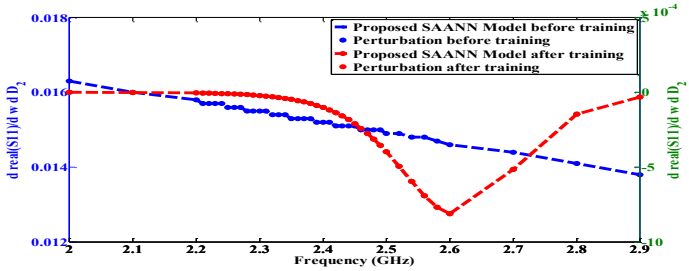


(c)

Figure 3.14: Derivative information of the real part of S_{11} to non-sensitivity variables S_1 and S_2 by the proposed SAANN model and perturbation sensitivity for $\frac{dRS_{11}}{dS_1}$ and $\frac{dRS_{11}}{dS_2}$ at (a) geometries #1, (b) geometries #2, and (c) geometries #3 for the coupled-line filter example. As shown in this figure, the proposed SAANN parametric model can predict the derivative information with respect to the geometrical variables, even though these variables are not available as sensitivity variables in original EM simulation.



(a)



(b)

Figure 3.15: Comparison of second-order derivatives of the real part of S_{11} to variables D_1 or D_2 and ANN weights (a) $\frac{d^2 real(S_{11})}{dw_{11}^2 dD_1}$, (b) $\frac{d^2 real(S_{11})}{dw_{11}^3 dD_2}$ versus frequency at geometry #1 before and after ANN training. Good agreement is observed between the proposed SAANN technique and EM perturbation techniques regardless whether the ANN is trained or not.

3.4.2 Parametric Modeling of a Junction

In this example, the proposed SAANN technique is applied to develop the parametric model of a family of junctions as shown in Fig. 3.16, where g is the gap distance between two conductive walls, d_h is the height of the tuning cylinder, and d_r is the radius of the tuning cylinder.

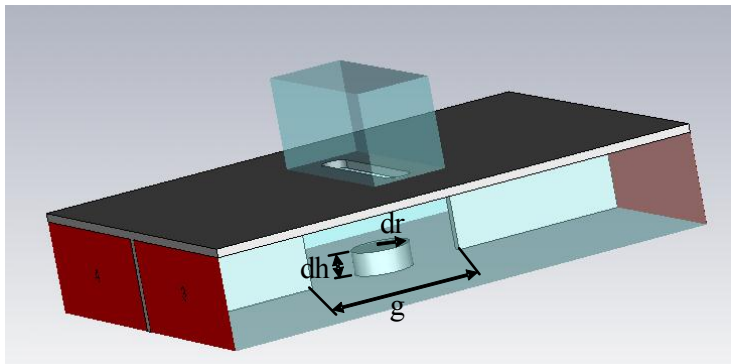


Figure 3.16: Structure of a junction and geometrical parameters used for generating training data for parametric modeling example (3D structure).

The structure of the proposed SAANN parametric model for the junction example is shown in Fig. 3.17. This parametric model has four inputs i.e., $x = [g\ d_h\ d_r\ \omega]^T$, which include three geometrical variables g d_h and d_r defined in Fig. 3.16 and frequency ω. In this example, g, d_h and d_r are all set as the sensitivity variables. This SAANN model combining the original and adjoint neural networks used for training has 40 outputs, i.e., $[RS_{11}\ IS_{11}\ RS_{21}\ IS_{21}\ RS_{31}\ IS_{31}\ RS_{41}\ IS_{41}\ \frac{dRS_{11}}{dg}\ \frac{dRS_{11}}{dd_h}\ \frac{dRS_{11}}{dd_r}\ \frac{dRS_{11}}{d\omega}\ \frac{dIS_{11}}{dg}\ \frac{dIS_{11}}{dd_h}\ \frac{dIS_{11}}{dd_r}\ \frac{dIS_{11}}{d\omega}\ ...\ \frac{dIS_{41}}{d\omega}]^T$ which are real and imaginary parts of S_{11} S_{21} S_{31} and S_{41}, the derivatives of real and imaginary parts of S_{11} S_{21} S_{31} and S_{41} with respect to four input variables (including frequency). The sensitivity analysis in CST EM simulator is performed to obtain the derivatives of real and imaginary parts of S_{11} S_{21} S_{31} and S_{41} to three sensitivity variables. Since frequency ω is not sensitivity-variables in EM simulation, the corresponding outputs from SAANN parametric model are left as free variables in the model training process. This is

achieved by setting the training weights for $[\frac{dRS_{11}}{d\omega}\ \frac{dIS_{11}}{d\omega}\ \frac{dRS_{21}}{d\omega}\ \frac{dIS_{21}}{d\omega}\ \frac{dRS_{31}}{d\omega}\ \frac{dIS_{31}}{d\omega}\ \frac{dRS_{41}}{d\omega}\ \frac{dIS_{41}}{d\omega}]^T$ as zero in our training program. The frequency range is from 7 GHz to 9 GHz with a step size of 6 MHz. The data range of training data and testing data is defined in Table 3.3. Partial orthogonal design of experiments method is also used to determine training and testing data.

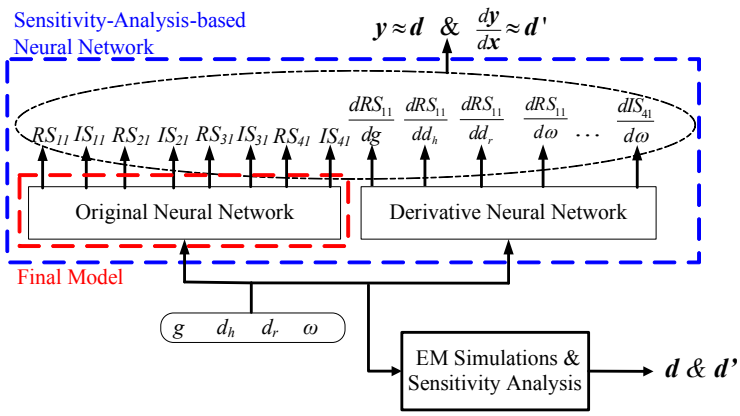


Figure 3.17: Structure of the proposed SAANN parametric model for the junction example.

Table 3.3: Definition of Training and Testing Data for Junction Example

Parameters		Training data			Testing data		
		Min	Max	Step	Min	Max	Step
Sensitivity Variables	g (mm)	16	24	1	16.5	23.5	1
	d_h (mm)	1.5	3.5	0.2	1.6	3.4	0.2
	d_r (mm)	2	4	0.2	2.1	3.9	0.2

Table 3.4 shows final results of training in terms of the average training and testing error of final trained model and its comparison with conventional ANN model

which was trained without using EM derivative data. Two sets of training data were used in order to compare the effect of training with respect to different sizes of training data. One set of training data has 80 samples (i.e., training with more training data), and another set has only 15 samples (i.e., training with less training data). From this table, we can see that with more training data, the conventional ANN model (i.e., trained without sensitivity information) can obtain a small training error and a consistent testing error. With less training data, conventional ANN model trained without sensitivity information can obtain a small training error but a larger testing error since the limited training data could not adequately represent the whole EM behavior of the original modeling problem. In contrast, the proposed SAANN parametric model can obtain a small training error and a small testing error with the same number of training data using sensitivity information to training neural networks. This technique introduced a new way to decrease the necessary training data in the model training process.

The SAANN model development cost for this junction example, including training data generation time (15 sets of training data) and model training time, is about 9.4 hours and for the conventional ANN model development (80 sets of training geometries) is about 45.7 hours. This further demonstrates that using the proposed technique, we speedup the model development time. Note that the training is a one-time investment, and the benefit of using the model accumulates when the model is used over and over again.

Fig. 3.18 depicts the outputs of the proposed SAANN parametric model for three different junction geometries #1, #2, and #3, and its comparison with EM

Table 3.4: Training and Testing Results for Junction Example

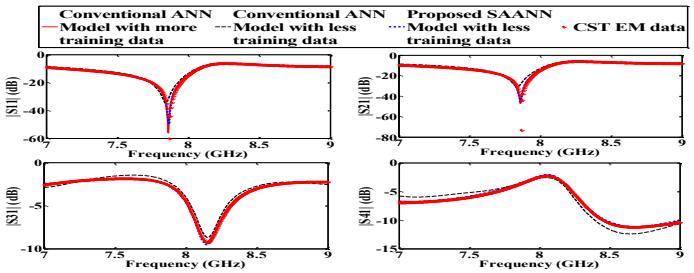
Model Type	Original Neural Network Structure	Average Training Error	Average Testing Error
Conventional ANN Model using 80 sets of training data	4-20-8	0.413%	0.473%
Conventional ANN Model using 15 sets of training data	4-20-8	0.482%	0.862%
Proposed SAANN Model using 15 sets of training data	4-20-8	0.453%	0.531%

data and conventional ANN model trained with training data of different sizes. The geometrical variables for three junctions are as follows:

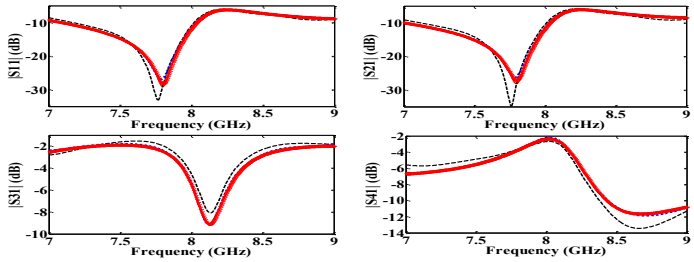
Geometry 1: g= 19.5 mm, d_h= 1.8 mm, d_r= 2.7 mm,

Geometry 2: g= 20.5 mm, d_h= 2.8 mm, d_r= 3.1 mm,

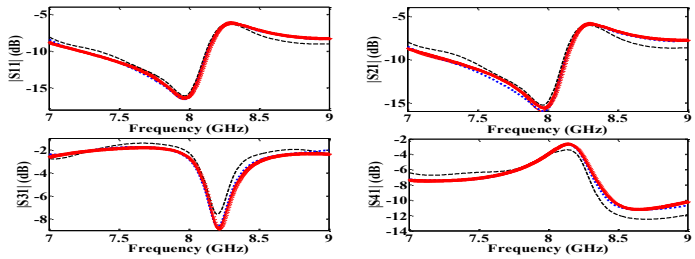
Geometry 3: g= 22.5 mm, d_h= 3.0 mm, d_r= 3.7 mm,



(a)



(b)



(c)

Figure 3.18: Comparison of the magnitude in dB of S_{11} S_{21} S_{31} and S_{41} of the proposed SAANN model, CST EM data and conventional ANN model with less or more training data for three different geometries (a) #1, (b) #2, and (c) #3 for the junction example. As shown in this figure, the proposed technique obtains more accurate model with less training data than conventional ANN technique. The match between proposed SAANN with original EM data is good even though the testing geometries used in the figures are never used in training.

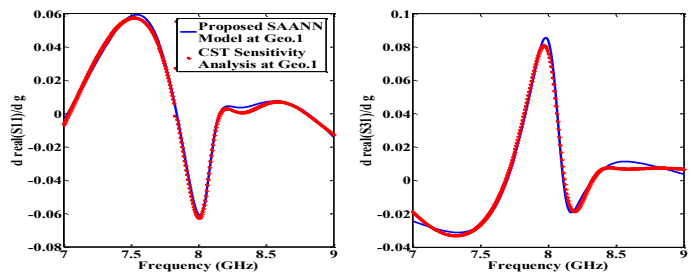
As shown in Fig. 3.18, broadband accuracy of the proposed SAANN parametric model is confirmed by its good agreement with EM data in terms of S_{11} S_{21}, S_{31} and S_{41} even these geometries are never used in the training process.

Table 3.5 also compares the proposed SAANN models and CST EM simulations in terms of CPU time for evaluating 100 different testing geometries of junction. As shown in Table 3.5, the trained ANN model is much faster than EM simulations.

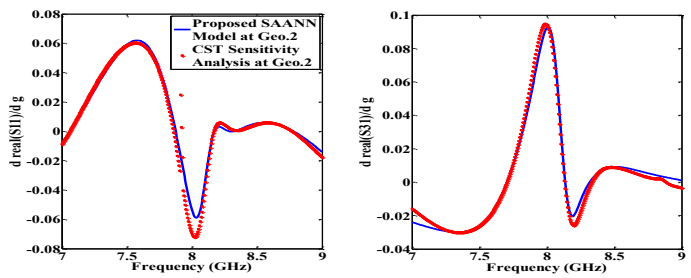
Table 3.5: CPU time of evaluating 100 different testing geometries for junction example.

Model Evaluation Type Different Testing Geometries	CPU Time of Evaluating 100
CST EM Simulations	95 minutes
Proposed SAANN Model	2.8s
Speedup Factor	2035

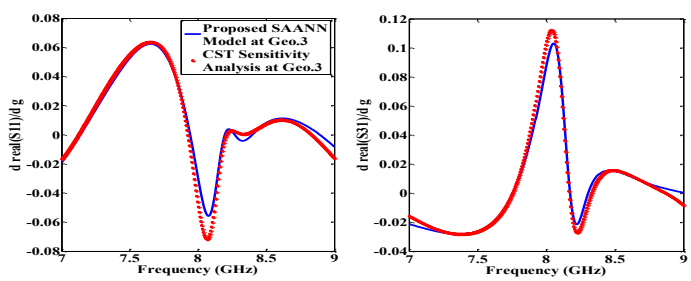
Here, we show another benefit of this technique where the trained model can accurately predict the derivative information of the junction responses with respect to geometrical variables. As shown in Fig. 3.19, we provide the comparison of derivatives of the real part of S_{11} and S_{31} with respect to sensitivity variables g by the proposed SAANN parametric model and CST sensitivity analysis at geometries #1, #2, and #3, respectively. As shown in this figure, the proposed SAANN model can accurately predict the derivative information, which is close to those obtained from CST sensitivity analysis, even though such geometries are never used in training process.



(a)



(b)



(c)

Figure 3.19: Comparison of the derivative information of the real part of S_{11} and S_{31} to sensitivity variable g by the proposed SAANN model and CST sensitivity analysis for $\frac{dRS_{11}}{dg}$ and $\frac{dRS_{31}}{dg}$ at (a) geometries #1, (b) geometries #2, and (c) geometries #3 for the Junction example. As shown in this figure, the proposed SAANN model can accurately predict the derivative information, even though such geometry is never used in the training process.

3.4.3 Parametric Modeling of a Cavity Filter

In this example, the proposed SAANN technique is applied to develop the parametric model of a family of microwave cavity filters as shown in Fig. 3.20, where H_{c1}, H_{c2}, and H_{c3}, which represent the heights of the cylinders respectively, are responsible for tuning the frequencies in the cavity, positioned at the cavity centers.

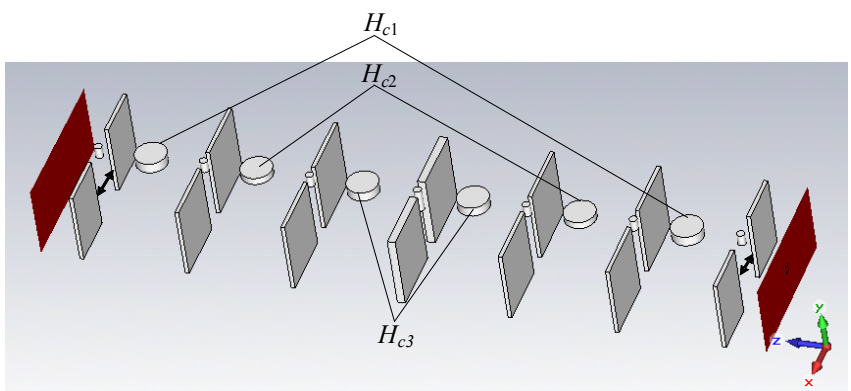


Figure 3.20: Structure of a microwave cavity filter and geometrical parameters used for generating training data for parametric modeling example (3D structure).

The structure of the proposed SAANN parametric model for the cavity filter example is shown in Fig. 3.21. The structure of the SAANN parametric model for microwave cavity filters is shown in Fig. 3.20. In this example, this SAANN parametric model has 4 inputs, i.e., $x = [H_{c1},\ H_{c2},\ H_{c3}\ \omega]^T$ which include three geometrical variables H_{c1}, H_{c2}, and H_{c3} defined in Fig. 3.20 and frequency ω. In this example, all three input geometrical variables are all set as the sensitivity variables. This SAANN model combining the original and adjoint neural networks used for training has 20 outputs, i.e., $[RS_{11}\ IS_{11}\ RS_{12}\ IS_{12}\ \frac{dRS_{11}}{dH_{c1}}\ \frac{dRS_{11}}{dH_{c2}}\ \frac{dRS_{11}}{dH_{c3}}\ \frac{dRS_{11}}{d\omega}$

$\frac{dIS_{11}}{dH_{c1}}$ ... $\frac{dIS_{12}}{d\omega}]^T$, which are the derivatives of real and imaginary parts of S_{11} and S_{12} to 4 inputs (including frequency). The sensitivity analysis in CST EM simulator is performed to obtain the derivatives of real and imaginary parts of S_{11} and S_{12} to three sensitivity variables. Since the frequency ω is not sensitivity-variables in CST EM simulation, the corresponding outputs from SAANN parametric model are left as free variables in the model training process. This is achieved by setting the training weights for $[\frac{dRS_{11}}{d\omega}\ \frac{dIS_{11}}{d\omega}\ \frac{dRS_{12}}{d\omega}\ \frac{dIS_{12}}{d\omega}]^T$ as zero in our training program. The frequency range is from 0.65 GHz to 0.7 GHz with a step size of 1.5 MHz. The data range of training data and testing data is defined in Table 3.6. Partial orthogonal design of experiments method is used to determine the size of training and testing data.

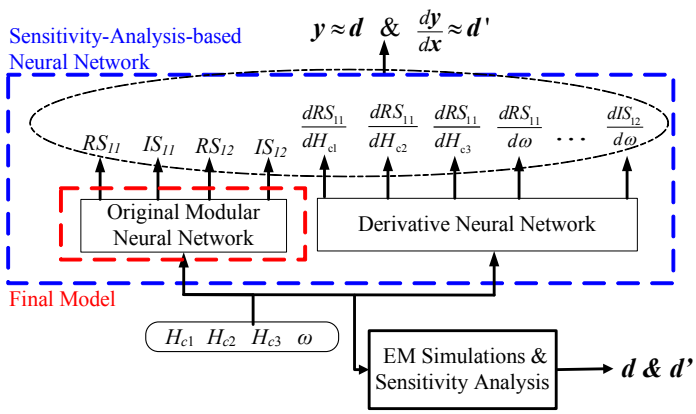


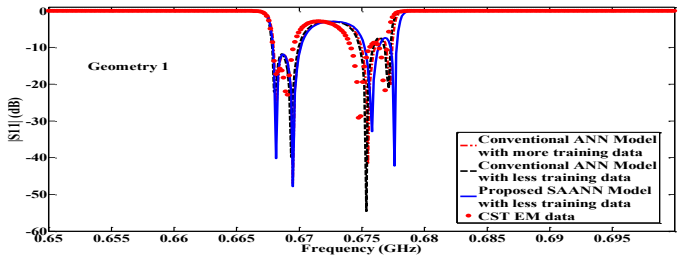
Figure 3.21: Structure of the proposed SAANN parametric model for the cavity filter example.

Table 3.7 shows final results of training in terms of the average training and testing error of final trained model and its comparison with conventional ANN model

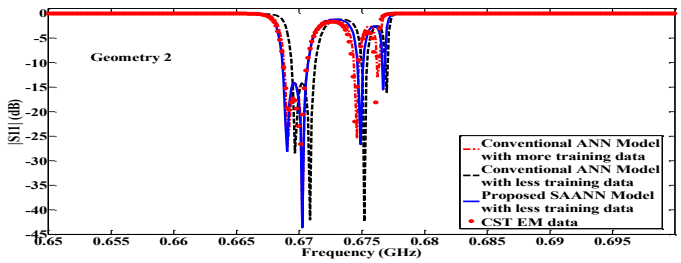
Table 3.6: Definition of Training and Testing Data for Cavity Filter Example

Parameters		Training data			Testing data		
		Min	Max	Step	Min	Max	Step
Sensitivity Variables	H_{c1} (mm)	25	29	1	25.5	28.5	1
	H_{c2} (mm)	16.5	20.5	1	17	20	1
	H_{c3} (mm)	23	27	1	23.5	26.5	1

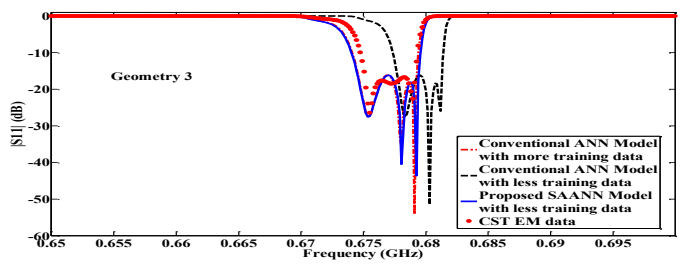
which was trained without using EM derivative data. Two sets of training data were used in order to compare the effect of training with respect to different sizes of training data. One set of training data has 120 samples (i.e., training with more training data), and another set has only 50 samples (i.e., training with less training data). From this table, we can see that with more training data, conventional ANN model (i.e., trained without sensitivity information) can obtain a small training error and a small testing error. With less training data, conventional ANN model trained without sensitivity information cannot obtain small testing error even though the training error is small. In contrast, the proposed SAANN model can obtain a small training error and a small testing error even with less training data. This is because that the SAANN technique incorporates not only the input-output behavior of the modeling problem, but also the derivative information from sensitivity analysis into the model training process. Therefore, using sensitivity information, we can obtain accurate model using less training data than without using sensitivity information.



(a)



(b)



(c)

Figure 3.22: Comparison of the magnitude in dB of S_{11} of the proposed SAANN model, CST EM data and conventional ANN model with less or more training data for three different geometries (a) #1, (b) #2, and (c) #3 for the cavity filter example. As shown in this figure, the proposed technique obtains more accurate model with less training data than conventional ANN technique. The match between proposed SAANN with original EM data is good even though the testing geometries used in the figures are never used in training.

Table 3.7: Training and Testing Results for Cavity Filter Example

Model Type	Original Neural Network Structure	Average Training Error	Average Testing Error
Conventional ANN Model using 120 sets of training data	4-25-20-8	1.17%	1.71%
Conventional ANN Model using 50 sets of training data	4-25-20-8	1.15%	5.41%
Proposed SAANN Model using 50 sets of training data	4-25-20-8	1.47%	1.59%

The SAANN model development cost for this cavity filter example, including training data generation time (50 sets of training data) and model training time, is about 17.2 hours and for the conventional ANN model development (120 sets of training geometries) is about 37.5 hours. This further demonstrates that using the proposed technique, we speedup the model development time. Note that the training is a one-time investment, and the benefit of using the model accumulates when the model is used over and over again.

Fig. 3.22 depicts the outputs of the proposed SAANN parametric model for three different filter geometries #1, #2, and #3, and its comparison with CST EM data and conventional ANN model trained with training data of different sizes. The geometrical variables for three filters are as follows:

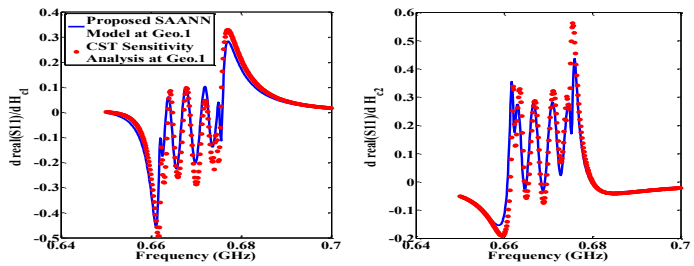
Geometry 1: H_{c1}=27 mm, H_{c2}=18.9 mm, H_{c3}=25 mm;

Geometry 2: H_{c1}=28.8 mm, H_{c2}=19.3 mm, H_{c3}=25.6 mm;

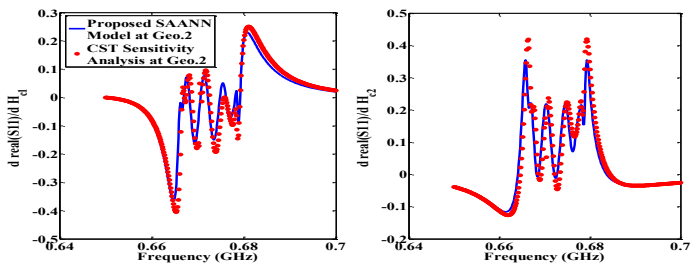
Geometry 3: H_{c1}=27.8 mm, H_{c2}=17.5 mm, H_{c3}=24 mm.

As shown in Fig. 3.22, broadband accuracy of the proposed SAANN parametric model is confirmed by its good agreement with EM data in terms of S_{11} even these geometries are never used in the training process.

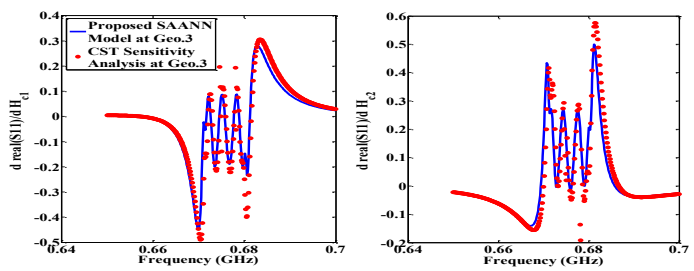
Here, we further show that the trained model can accurately predict the derivative information with respect to geometrical variables. As shown in Fig. 3.23, we provide the comparison of the derivatives real part of S_{11} to sensitivity variables H_{c1}, H_{c2}, and H_{c3} by the proposed SAANN parametric model and CST sensitivity analysis at geometries #1, #2, and #3, respectively. As shown in this figure again, the proposed SAANN parametric model can accurately predict the derivative information, which is much closer to those obtained from CST sensitivity analysis, even though such geometries are never used in training process.



(a)



(b)



(c)

Figure 3.23: Comparison of the derivative information of the real part of S_{11} to sensitivity variables H_{c1}, H_{c2} by the proposed SAANN parametric model and CST sensitivity analysis for $\frac{dRS_{11}}{dH_{c1}}$, and $\frac{dRS_{11}}{dH_{c2}}$ at (a) geometries #1, (b) geometries #2, (c) geometries #3 for the cavity filter example. As shown in this figure, the proposed SAANN model can accurately predict the derivative information, which are much closer to those obtained from CST sensitivity analysis, even though such geometry is never used in the training process.

3.5 Summary and Conclusion

In this chapter, a novel sensitivity-analysis-based adjoint neural network technique for developing parametric model of microwave passive components has been presented. Using sensitivity information, this technique introduces a new way to develop accurate neural network models with less EM data than that without using sensitivity. The parametric SAANN models are well suited for the purpose of establishing EM component libraries, where the trained models can be re-used again and again for microwave passive components design with different specifications. The SAANN technique can also provide sensitivity information with respect to geometric parameters which are not sensitivity variables in the original EM simulator. Therefore the method also helps to extend sensitivity analysis beyond the variable-limits in EM simulators.

While SAANN has its advantages over conventional methods it is restricted to static cases where no time-domain transient data is present. The next chapter presents a technique that extends the concepts present in SAANN for application to cases where transient data is used to develop dynamic models.

Chapter 4

Adjoint State-Space Dynamic Neural Network Technique for Nonlinear Microwave Electronic/Photonic Component Modeling

In this chapter an adjoint state-space dynamic neural network (ASSDNN) method for modeling nonlinear circuits and components is presented. This method is used for modeling transient behavior of nonlinear electronic and photonic components. The proposed technique is an extension of the existing state-space dynamic neural network technique (SSDNN). The new method simultaneously adds derivative information to the training patterns of nonlinear components allowing the training to be done with less data without sacrificing model accuracy and consequently makes training faster and more efficient. Also, this method has been formulated such that it can be suitable for parallel computation. The use of derivative information and parallelization make training using the proposed technique much faster

than SSDNN. In addition, the models created using the proposed method are much faster to evaluate compared to conventional models present in traditional circuit simulation tools. The validity of the proposed technique is demonstrated through transient modeling of physics-based CMOS driver, commercial NXP's 74LVC04A inverting buffer and nonlinear microwave photonic components.

4.1 Introduction

In the past few years artificial neural networks (ANNs) have gained attention as a valuable computer aided design (CAD) tool for modeling high frequency circuits in the microwave area [55], [65]. The recently introduced state-space dynamic neural networks (SSDNN) can be seen as a generalized form of DNN-based methods.

In the present chapter, a further advance over the SSDNN technique titled adjoint state-space dynamic neural network (ASSDNN), is developed and discussed. Similar to conventional SSDNN, the proposed technique can train and model the input-output relationship of a nonlinear component/circuit without having to rely on the internal details of the block. In addition, the ASSDNN method also uses derivatives of the output waveforms as the training data. As a result, the training associated with ASSDNN is more efficient and requires less training data compared to conventional SSDNN. The concept of using derivative information in training was introduced in [103] for ANNs; this chapter extends this concept for efficient training of DNNs where the inputs and outputs are time-domain waveforms. Further, ASSDNN was developed so as to utilize the advantage of parallel computation on multiple cores/processors available in present day microprocessors. This provides

an additional speed-up compared to what is already obtained with the use of derivative information which together enables ASSDNN to provide a significant efficiency improvement in nonlinear component/circuit modeling.

In order to demonstrate the accuracy and efficiency of the proposed method, in this chapter, ASSDNN was applied to model microwave photonic and physics-based components for use in SPICE-like simulators. With the continuous increase in the speed and frequency of signal, signal integrity is becoming more important in VLSI/Electronic circuits. Developing fast and accurate models for nonlinear behaviors of driver/receiver buffers are the key to signal integrity based design of high-speed interconnects with nonlinear terminations [97], [104]-[106]. Type of models could be behavioral such as input and output buffer information specification (IBIS) models [107]-[110], transistor-level models [98],[109], or physics-based models. Evaluating the transient behavior of nonlinear electronic circuits such as drivers using physics-based models requires time-consuming computations. When repetitive evaluations of the circuit are needed, it makes the calculations very costly. This necessitates the development of a more efficient and accurate computational form for building models for nonlinear electronics circuits to replace either their original detailed electromagnetic (EM)/physics models in order to speed up microwave design [111], [112], or their simplified behavioral models in order to increase the model accuracy.

Also, modeling photonic components has garnered much attention in the recent past owing to technological advances that enabled the inclusion of photonic components at the microelectronic level leading to the co-existence of electronic and

microwave photonic components at the same level of design hierarchy [113]-[122]. Simulation frameworks such as OptiSPICE [113] have been introduced to address co-simulation of microwave photonic and electronic components within the same transient engine. However, models for components such as nonlinear waveguides in transient simulators such as OptiSPICE still rely on the Split-Step Fourier (SSF) method [123] uses frequency domain extensively. As such simulating electronic-photonic circuits that consist of nonlinear waveguides would resort to expensive time-domain convolutions to combine the response of these components.

The aforementioned problems regarding electrical-optical modeling are addressed in this chapter by developing time-domain models for microwave electronic circuits, nonlinear waveguides and nonlinear ring-resonators using ASSDNN.

This chapter is organized as follows. Section 4.2 discusses the conventional state-space dynamic neural network (SSDNN) followed by Section 4.3 which presents the proposed method and discusses its details that include utilization of derivative information during training and parallel implementation. In Section 4.4 the proposed method is applied to four different photonic-electronic systems where time domain models of nonlinear electronic circuit and microwave photonic elements are developed using the proposed technique and compared with existing techniques such as conventional SSDNN, OptiSPICE, MINIMOS-NT and IBIS to model and simulate these elements. The conclusions are finally presented in Section 4.5.

4.2 The Conventional SSDNN Nonlinear Modeling Structure

4.2.1 General Structure

The goal here is to develop a model with similar input-output relationship as the original complex nonlinear modeling problem with an acceptable error range. At the same time evaluation of the model should be faster than that of the original model. Suppose the model is represented by $M : u(t) \rightarrow y(t)$, where $u(t)$ is a vector of size M which includes M transient input signals of a nonlinear circuit (voltages/currents etc.) and $y(t)$ is a vector of size K including the K transient output signals of the same circuit. Based on the state-space concept introduced in [98] and [100], the general SSDNN equations that can model the original nonlinear circuit, but with lesser complexity and much faster computation time, is formulated as follows,

$$\begin{aligned} \dot{x}(t) &= -x(t) + \tau g_{ANN}(u(t), x(t), w) \\ y(t) &= Cx(t) \end{aligned} \tag{4.1}$$

where $x(t)$ is a vector of size N containing state variables $(x_1(t), x_2(t), \ldots, x_N(t))$, and $g_{ANN}(t)$ a vector of size N including the outputs of a feed-forward multilayer perceptron (MLP) $(g_{ANN-1}(t), g_{ANN-2}(t), ..., g_{ANN-N}(t))$ [55] that has $M+N$ input neurons and 1 hidden layer with H hidden neurons. W is the matrix of the weight parameters of this MLP and $C[K \times N]$ is the output matrix that maps the state variables to the output variables.

For simplifying the calculations, the weight matrix (W), is divided into 3 matrices as described in [98]. W_u contains the weights connecting inputs ($u(t)$) to the

hidden neurons of the hidden layer,

$$W_u = \begin{bmatrix} w_{11}^2 & w_{12}^2 & \cdots & w_{1M}^2 \\ w_{21}^2 & w_{22}^2 & \cdots & w_{2M}^2 \\ \vdots & \vdots & \cdots & \vdots \\ w_{H1}^2 & w_{H2}^2 & \cdots & w_{HM}^2 \end{bmatrix}.$$

W_s contains the weights connecting state variables ($x(t)$) to the hidden neurons of the hidden layer,

$$W_s = \begin{bmatrix} w_{1,M+1}^2 & w_{1,M+2}^2 & \cdots & w_{1,M+N}^2 \\ w_{2,M+1}^2 & w_{2,M+2}^2 & \cdots & w_{2,M+N}^2 \\ \vdots & \vdots & \cdots & \vdots \\ w_{H,M+1}^2 & w_{H,M+2}^2 & \cdots & w_{H,M+N}^2 \end{bmatrix}$$

and W_o contains the weights connecting hidden neurons of the hidden layer to the outputs of the MLP ($y(t)$),

$$W_o = \begin{bmatrix} w_{11}^3 & w_{12}^3 & \cdots & w_{1H}^3 \\ w_{21}^3 & w_{22}^3 & \cdots & w_{2H}^3 \\ \vdots & \vdots & \cdots & \vdots \\ w_{N1}^3 & w_{N2}^3 & \cdots & w_{NH}^3 \end{bmatrix}$$

where w_{ij}^l is the weight between the i_{th} neuron of the l_{th} layer and the j_{th} neuron of the $(l-1)_{th}$ layer. Using W_u, W_s, and W_o (4.1) can be rewritten as

$$\dot{x}(t) = -x(t) + \tau W_o \sigma(W_u u(t) + W_s x(t)) \tag{4.2}$$

where $\sigma(.)$ is a function of size H of nonlinear activation functions of the hidden neurons in the hidden layer of the MLP. They are assumed to be bounded and monotonically increasing. The sigmoid and hyperbolic tangent functions are among the most commonly used activation functions. Figure 4.1 shows the detailed structure of the 3-layer MLP used in the conventional SSDNN technique.

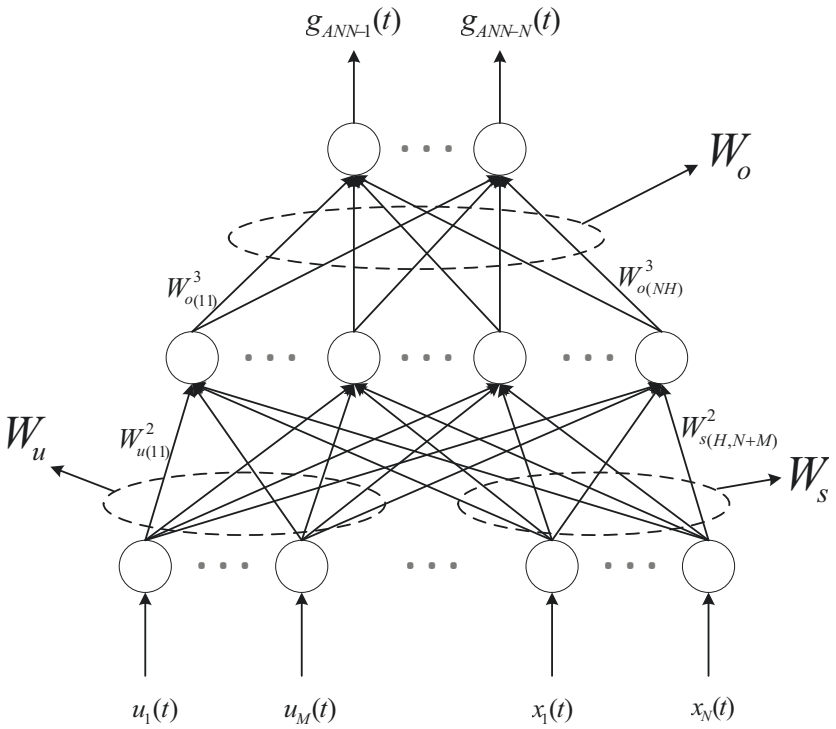


Figure 4.1: Structure of the MLP used in SSDNN. Inputs of the MLP include 2 parts: $u(t)$ and $x(t)$. The outputs ($g_{ANN}(t)$) are the same number as the state variables.

In addition to the state-space equations, another set of equations, called adjoint

state-space equations of SSDNN, are defined in [98] as

$$\dot{\hat{x}}(t) = \hat{x}(t) - \tau {W_s}^T G(t) {W_o}^T \hat{x}(t) + C^T(y^m(t) - y_d^m(t)) \tag{4.3}$$

where $G(t)$ is

$$G(t) = diag\left[\sigma_1'\left(W_u^{(1)}u(t) + W_s^{(1)}x(t)\right), \ldots, \sigma_H'\left(W_u^{(H)}u(t) + W_s^{(H)}x(t)\right)\right] \tag{4.4}$$

with σ' being the derivative of the activation function (σ) and ${W_u}^{(i)}$ and ${W_s}^{(i)}$ being the i_{th} rows of W_u and W_s respectively. The boundary condition for (4.3) is assumed to be $\hat{x}(T) = 0$ where T is a large number and is practically close to infinity for the purpose of this problem. The time-domain solution for (4.4) is obtained by solving this set of differential equations backward in time from $t = T$ to $t = 0$.

4.2.2 Training of the Conventional Model

Assume S is the total number of input transient waveforms obtained from the circuit which will be used for training. Also assume $u_d^m(t)$ and $y_d^m(t)$ are the m_{th} input and output training waveforms respectively for the time interval $[0, T]$. $y^m(t)$ is the output obtained by the model corresponding to the output $y_d^m(t)$. For minimizing the difference between the SSDNN model output ($y^m(t)$) and the original output data ($y_d^m(t)$) an error function has been defined as

$$E_d = \sum_{m=1}^{S} E_d^m \tag{4.5}$$

where E_d^m is the error for the m_{th} training waveform and is calculated as

$$E_d^m = \frac{1}{2}\int_0^T \left\| y^m(t) - y_d^m(t) \right\|^2 dt \tag{4.6}$$

In order to train the SSDNN model we form a constrained optimization problem with the objective function to be minimized as E_d with the equations in (4.1) as its constraints. Solving this optimization problem results in optimum values for the weights W_u, W_s, and W_o which result in the minimum value for E_d. If gradient-based techniques are used to solve the optimization problem, derivative information of the objective function (in this case the error function E_d) is required with respect to design variables (i.e. weights and the elements of the C matrix). These derivatives are also called sensitivities and these sensitivities as calculated in [98] as,

$$\frac{dE_d}{dw_{ij}^l} = \sum_{m=1}^{S} \frac{dE_d^m}{dw_{ij}^l} \tag{4.7}$$

and

$$\frac{dE_d}{dc_{ij}} = \sum_{m=1}^{S} \int_0^T (y_i^m - y_{di}^m) x_j dt \tag{4.8}$$

where, $\frac{dE_d^m}{dw_{ij}^l}$ can be evaluated as

$$\frac{dE_d^m}{dw_{ij}^l} = -\int_0^T \hat{x}\left[\tau \frac{dW_o}{dw_{ij}^l}\sigma + \tau W_o G\left(\frac{dW_u}{dw_{ij}^l}u + \frac{dW_s}{dw_{ij}^l}x\right)\right] dt \tag{4.9}$$

and y_i^m and y_{di}^m are the i_{th} output of the model and training data for the m_{th} training waveform respectively.

4.3 The Proposed Method

In this section a new method titled adjoint state-space dynamic neural network is proposed that includes derivative information during the training process that renders the training more efficient than traditional techniques. This section is organized as follows: Sub-section 4.3.1 discusses the structure of the proposed dynamic neural network followed by sub-section **??** that discusses the stability properties of the models obtained from ASSDNN. Sub-section 4.3.2 presents implementation details concerning parallelization of the proposed technique.

4.3.1 The Adjoint State-Space Dynamic Neural Network Structure

The concept inspiring development of the proposed method is based on the use of derivative information of the output in the training process which provides more information for the algorithm during training and makes the training easier. In conventional SSDNN training data includes input/output information of the component; in the proposed method training data not only includes input/output information of the component but also includes derivatives of the transient responses of the output. This concept was applied to conventional neural networks for parametric modeling in [103] with success. In this chapter this concept is applied for the first time to dynamic neural networks for time-domain modeling. Results show that the proposed method requires less training data to get the same accuracy compared to conventional methods owing chiefly to the use of derivative information during training. It can theoretically be shown that the error resulting from the model

obtained from ASSDNN would be less than that resulting from a model obtained using SSDNN.

Lemma 1. *For a certain nonlinear circuit let $f_0(t)$, $f_1(t)$, and $f_2(t)$ be the original transient output signal, output of the model obtained using conventional SSDNN method, and output of the model obtained using the proposed ASSDNN method respectively. Let $E_1(t_0)$ and $E_2(t_0)$ be the training error of the SSDNN ($|f_1(t_0) - f_0(t_0)|$) and ASSDNN ($|f_2(t_0) - f_0(t_0)|$) models at the time point t_0 respectively. Then,*

$$\lim_{\Delta t \to 0} \frac{E_2(t_0 + \Delta t) - E_1(t_0 + \Delta t)}{\Delta t} \leq 0$$

Proof. Considering the first few terms in the Taylor series expansion of $f_0(t)$, $f_1(t)$, and $f_2(t)$ we have

$$f_0(t) = f_0(t_0) + f_0'(t_0) \cdot \Delta t + \sum_{i=2}^{n} \frac{1}{i!} f_0^{(i)}(t_0) \cdot \Delta t^{(i)}$$

$$f_1(t) = f_0(t_0) + E_1(t_0) + f'_1(t_0) \cdot \Delta t + \sum_{i=2}^{n} \frac{1}{i!} f_1{}^{(i)}(t_0) \cdot \Delta t^{(i)}$$

$$f_2(t) = f_0(t_0) + E_2(t_0) + f_0'(t_0) \cdot \Delta t + E_2'(t_0) \cdot \Delta t + \sum_{i=2}^{n} \frac{1}{i!} f_2^{(i)}(t_0) \cdot \Delta t^{(i)} \tag{4.10}$$

where $E_2'(t_0)$ is the training error between the derivative of the response of the proposed model $f_2'(t)$ and the derivative of training data $f_0'(t)$ at the time sample t_0.

Assuming that the training based on SSDNN and ASSDNN techniques are performed well, the training errors $E_1(t_0)$, $E_2(t_0)$, and $E_2'(t_0)$ can be taken to be 0. Neglecting $\Delta t^{(i)}$ for higher order parts of the equation for small Δt, the testing errors of the proposed and the conventional models for the sample time $t = t_0 + \Delta t$ can be calculated as follows,

$$E_1(t_0 + \Delta t) = |f_1(t) - f_0(t)| = \Bigg| [f_1{}'(t_0) - f_0{}'(t_0)] \cdot \Delta t + \sum_{i=2}^{n} \frac{1}{i!} \left[f_1{}^{(i)}(t_0) - f_0{}^{(i)}(t_0) \right] \cdot \Delta t^{(i)} \Bigg|$$

$$E_2(t_0+\Delta t) = |f_2(t) - f_0(t)| = \left| \sum_{i=2}^{n} \frac{1}{i!} \left[{f_2}^{(i)}(t_0) - {f_0}^{(i)}(t_0) \right] \cdot \Delta t^{(i)} \right|$$

This implies that

$$\lim_{\Delta t \to 0} \frac{E_2(t_0+\Delta t) - E_1(t_0+\Delta t)}{\Delta t} = -\,|{f_1}'(t_0) - {f_0}'(t_0)| \leq 0$$

□

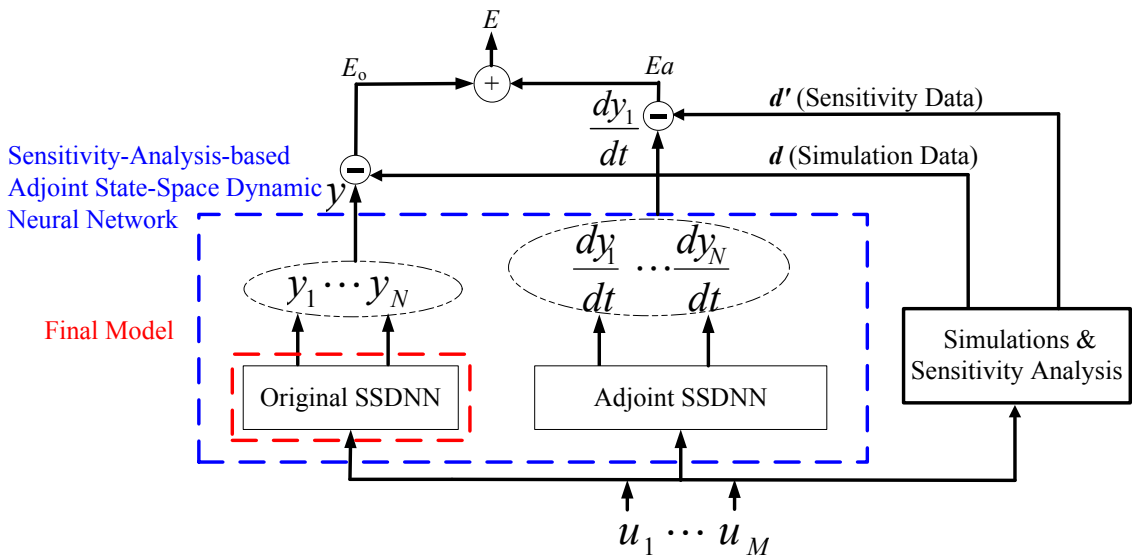


Figure 4.2: The structure of the proposed ASSDNN-based model. It includes two parts: original state-space dynamic neural network and the adjoint state-space dynamic neural network, where (u_1, ..., u_M) and (y_1, ..., y_N) represent the transient input and output signals of a nonlinear circuit respectively.

This shows that the testing error obtained from the model trained by the proposed technique using derivative information is always less than the testing error obtained from the model trained by the conventional SSDNN method.

Using the same nomenclature as in (4.1) ASSDNN equations can be formulated

as

$$\begin{aligned}
\dot{x}(t) &= -x(t) + \tau g_{ANN}\left(u(t), x(t), w\right) \\
y(t) &= Cx(t) \\
\dot{y}(t) &= C\dot{x}(t).
\end{aligned} \tag{4.11}$$

The matrix of weights W is again sub-divided into three matrices W_u, W_s, and W_o as explained in Section 4.2. Following the procedure in Section 4.2 $\dot{x}(t)$ for ASSDNN can be written as

$$\dot{x}(t) = -x(t) + \tau W_o \sigma\left(W_u u(t) + W_s x(t)\right) \tag{4.12}$$

and as such $\dot{y}(t)$ can be written as

$$\dot{y}(t) = -y(t) + \tau C W_o \sigma(W_u u(t) + W_s x(t)). \tag{4.13}$$

Compared to SSDNN formulation the ASSDNN formulation has the derivative of the output involved and training with the use of derivatives makes modeling using ASSDNN more efficient when compared to SSDNN as shown by Lemma 1. The structure of an ASSDNN-based model is graphically shown in Figure 4.2. When ASSDNN is applied for the purpose of modeling optical and optical-electrical components the inputs and outputs $x(t)$ and $y(t)$ could represent either voltages/currents in the electrical part of the component or the magnitude/phase of the electromagnetic field present in the optical part of the component.

Training of the ASSDNN-based model is achieved by solving an optimization problem formulated such that its solution minimizes the error between the response

generated from the ASSDNN-based model and the data obtained from transient simulations using SPICE-like simulators while satisfying the constrains described in (4.11). The objective function of this optimization problem is a function of the weights of the MLP and the elements of the C matrix and is given as (assuming similar variable names as defined in section 4.2),

$$E = \sum_{m=1}^{S} E^m \tag{4.14}$$

where E^m is the total training error of the ASSDNN-based model for the m_{th} training waveform and is calculated as

$$E^m = E_O^m + E_A^m \tag{4.15}$$

where E_O^m and E_A^m are the original and adjoint training errors for the m_{th} training waveforms respectively and are calculated as,

$$E_O^m = \frac{1}{2} K \int_0^T \|y^m - y_d^m\|^2 dt \tag{4.16}$$

and

$$E_A^m = \frac{1}{2} K' \int_0^T \|\dot{y}^m - \dot{y}_d^m\|^2 dt = \frac{1}{2} K' \int_0^T \|-y^m + \tau C W_o \sigma - \dot{y}_d^m\|^2 dt \tag{4.17}$$

where $y_d^m(t)$ and $\dot{y}_d^m(t)$ are the m_{th} output training waveform and its derivative for the time interval $[0,\ T]$ and $y^m(t)$ and $\dot{y}^m(t)$ are the output of the model based on ASSDNN technique and its derivative as calculated from (4.11). K and K' are appropriate scaling factors.

The objective function is further modified using Lagrangian functions [124] in order to incorporate constraints (4.11) of the optimization problem. For the mth waveform the modified objective function can be written as

$$L^m = L_O^m + E_A^m \tag{4.18}$$

where,

$$L_O^m = E_O^m + \hat{x}^T(t)[\dot{x}(t) + x(t) - \tau W_o \sigma \left(W_u u(t) - W_s x(t)\right)] \tag{4.19}$$

where, $\hat{x}(t)$ is a vector of time-dependent Lagrange parameters.

In addition, the use of gradient-based optimization techniques require sensitivity information of the objective function. The sensitivity of the objective function with respect to the weights of the MLP can be evaluated as

$$\begin{aligned}
\frac{dL^m}{dw_{ij}^l} = \int_0^T \Big[& -\dot{\hat{x}}^T + \hat{x}^T - \hat{x}^T \tau W_o G W_s + K(y^m - y_d^m)^T C + \\
& K'(y^m + \dot{y}_d^m - \tau C W_o \sigma)^T \left(C - \tau C W_o G W_s\right) \Big] \frac{dx}{dw_{ij}^l} dt \\
+ \hat{x}^T \frac{dx}{dw_{ij}^l} \Big]_0^T & - \int_0^T \left(\hat{x}^T + K'(y^m + \dot{y}_d^m - \tau C W_o \sigma)^T C \right) \\
& \times \left(\tau \frac{dW_o}{dw_{ij}^l} \sigma + \tau W_o G \left(\frac{dW_u}{dw_{ij}^l} u + \frac{dW_s}{dw_{ij}^l} x \right) \right) dt
\end{aligned} \tag{4.20}$$

c The first integral in (4.20) includes $\frac{dx}{dw_{ij}^l}$ which is difficult to evaluate. In order to circumvent this issue $\hat{x}$ is carefully chosen such that the coefficient of $\frac{dx}{dw_{ij}^l}$ in L^m vanishes. As such $\hat{x}$ should satisfy the equation

$$\dot{\hat{x}}(t) =$$

$$\hat{x}(t) - \tau {W_s}^T G(t) {W_o}^T \hat{x}(t) + KC^T \left(y^m(t) - y_d^m(t)\right) + \tag{4.21}$$

$$K' \left(C^T - \tau {W_s}^T G {W_o}^T C^T\right) \left(y^m(t) + \dot{y}_d^m(t) - \tau C W_o \sigma\right)$$

These are the adjoint equations of ASSDNN and $\hat{x}$ represents the adjoint state variables (the reason the proposed method is called adjoint SSDNN). Assuming the boundary condition $\hat{x}(T) = 0$ this equation can be solved by marching backward in time. Further, it should be noted that

$$\hat{x}^T \frac{dx}{dw_{ij}^l}\Bigg]_0^T = 0 \tag{4.22}$$

Using (4.21) and (4.22), (4.20) can be written as,

$$\begin{aligned} \frac{dL^m}{dw_{ij}^l} = &- \int_0^T \left(\hat{x}^T + K'(y^m + \dot{y}_d^m - \tau C W_o \sigma)^T C\right) \\ &\times \left(\tau \frac{dW_o}{dw_{ij}^l}\sigma + \tau W_o G \left(\frac{dW_u}{dw_{ij}^l} u + \frac{dW_s}{dw_{ij}^l} x\right)\right) dt \end{aligned} \tag{4.23}$$

Equation (4.23) can be further simplified based on the location of w_{ij}^l for the purpose of efficient evaluation as,

$$\frac{dL^m}{dw_{ij}^l} = \begin{cases} -\int_0^T \left(\hat{x}_i + K'(y^m + \dot{y}_d^m - \tau C W_o \sigma)^T C^{T(i)}\right) \\ \times \tau \sigma_j dt \quad \text{for } l = 3 \\ -\int_0^T \tau \left(\hat{x} + K'(y^m + \dot{y}_d^m - \tau C W_o \sigma)^T C\right) \\ \times {W_o}^{T(i)} \sigma'_i u_j dt \quad \text{for } 1 \leq j \leq M, l = 2 \\ -\int_0^T \tau \left(\hat{x} + K'(y^m + \dot{y}_d^m - \tau C W_o \sigma)^T C\right) \\ \times {W_o}^{T(i)} \sigma'_i x_{j-M} dt \quad \text{for } j > M, l = 2 \end{cases} \tag{4.24}$$

where $C^{T(i)}$ and $W_o^{T(i)}$ are the i_{th} rows of C^T and W_o^T respectively. Further, sensitivity of the modified objective function w.r.t c_{ij} can be evaluated as,

$$\begin{gathered} \frac{dL^m}{dc_{ij}} = K \int_0^T \left(y_i^m - y_{di}^m\right) x_j dt + \\ K' \int_0^T \left(-y_i^m - \dot{y}_{di}^m + \tau C^{(i)} W_o \sigma\right) \left(-x_j + \tau {W_o}^{(j)} \sigma\right) dt \end{gathered} \tag{4.25}$$

Finally, sensitivity of the overall modified objective function $L = \sum_{m=1}^{S} L^m$ can be computed using

$$\frac{dL}{dw_{ij}^l} = \sum_{m=1}^{S} \frac{dL^m}{dw_{ij}^l} \quad \text{and} \quad \frac{dL}{dc_{ij}} = \sum_{m=1}^{S} \frac{dL^m}{dc_{ij}} \tag{4.26}$$

Noteworthy to mention that, as the computation of derivatives are performed analytically using the proposed method, it does not include more data points to be generated. Also, if the training data used during training process are not accurate

enough, it can draw a limitation to the training technique.

Finally, the steps of the proposed training process in 1 iteration can be summarized as follows,

1. Calculation of the state variables, $x(t)$, and the outputs, $y(t)$, according to (4.1).

2. Calculation of $\hat{x}(t)$ according to (4.3).

3. Calculation of the derivatives $\frac{dE}{dw_{ij}^{l}}$ and $\frac{dE}{dc_{ij}}$ according to (4.23) and (4.25).

The block diagram in Figure 4.3 shows the flowchart of the proposed ASSDNN training technique in detail. After the completion of the training process, the results will be validated by a set of test waveforms. After verifying the accuracy of the model developed using the proposed method, the model can be incorporated into transient SPICE-like simulation tools.

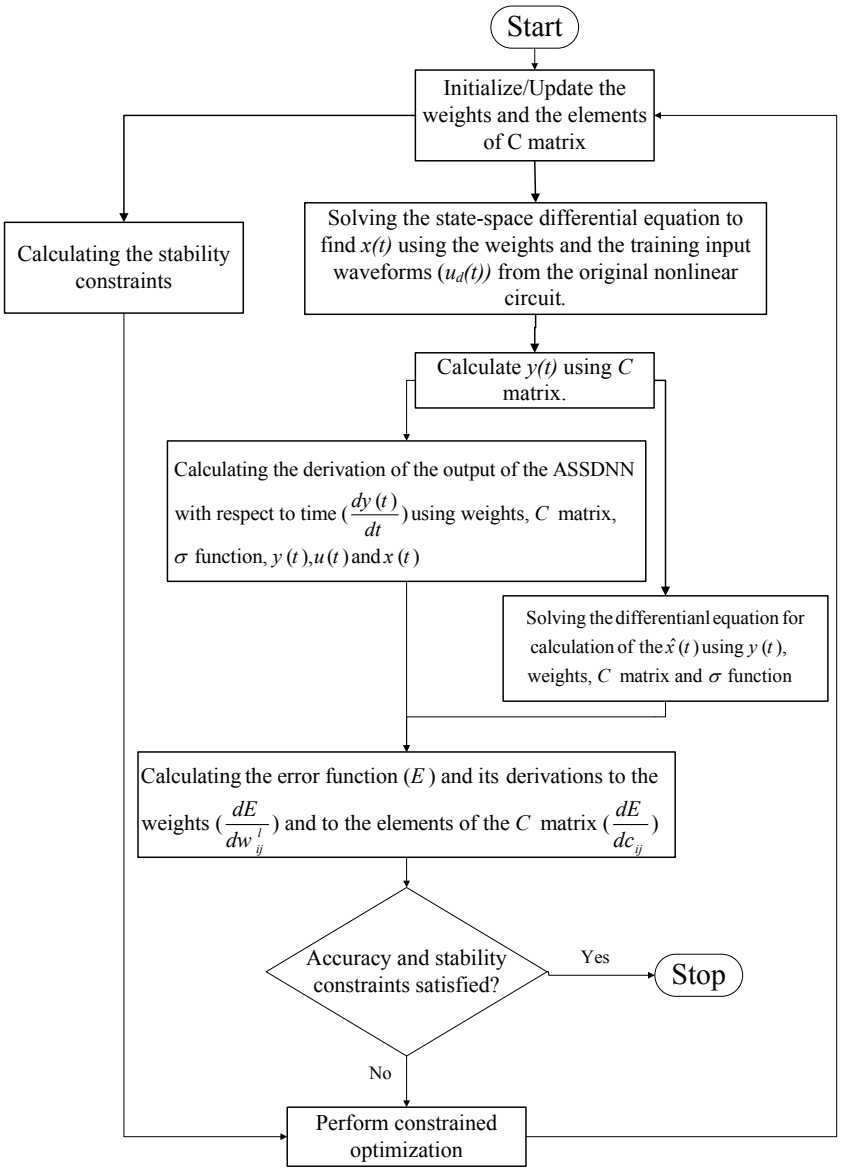


Figure 4.3: Block diagram describing the proposed adjoint state-space dynamic neural network (ASSDNN) training technique. As it can be seen, the derivatives are analytically calculated and passed to the optimizer to be used for the optimization process.

4.3.2 Parallel Computation

This sub-section details the method used to parallelize the training process of the proposed method. It should first be noted that the iterations involved in solving the fundamental optimization problem in training is sequential and cannot be parallelized. As such the opportunity to parallelize exists only in each iteration. Within each iteration there are three major computations:

1. Computation of the constraints
2. Computation of the objective function (error function)
3. Computation of the derivatives

The most time-consuming step involved in each iteration is the one related to derivative computation. Table 4.1 shows a comparison of the computation time between the three different steps for a state-space dynamic neural network with 15 hidden neurons and 10 state variables using a single core. As it can be seen in Table 4.1, the elapsed time of the derivative computations is more than the other parts (using a single core without parallelization). Therefore, the effort to parallelize training related to this method was focused on the derivative computation part.

Table 4.1: Comparison of the computation time between three major computation parts of the training process in a sample state-space dynamic neural network with 15 hidden neurons and 10 state variables using a single core

	Computation of the constraints	Computation of the objective function	Computation of the derivatives
Time	0.00008 (s)	0.166 (s)	2.2 (s)

The derivation computation in turn consists of four parts and are described below:

1. The derivative of the error function w.r.t. the elements of W_u matrix ($\frac{dE}{dW_u}$)
2. The derivative of the error function w.r.t. the elements of W_s matrix ($\frac{dE}{dW_s}$)
3. The derivative of the error function w.r.t. the elements of W_o matrix ($\frac{dE}{dW_o}$)
4. The derivative of the error function w.r.t. the elements of C matrix ($\frac{dE}{dC}$)

Each of the above parts contains derivatives of several elements. The computation of each of these elements can be performed independently without depending on the information from other elements. Taking advantage of this inherent parallelizable structure of the derivative computation a significant speed up can be obtained in the training process of the proposed method.

Table 4.2 shows a comparison between the training times of one iteration in the conventional training method using 1 core without parallelization and using several

cores in parallel. As it can be seen from the table, the training time is reduced significantly using more number of cores. Here, constrained training was performed using Matlab 7.10 with the Parallel Computing toolbox and $fmincon$ function from the optimization toolbox [125]. During the training process, weight parameters and the C matrix elements were initialized by uniform random distribution between [-1, 1] and they were used as starting points for the training. Also, one sample input and output training waveforms were used in the training process.

Table 4.2: Comparison between the training times of 1 iteration in the conventional training method using different number of cores

	1 core	2 cores	4 cores	8 cores
1 iteration training time (s)	8.6	5.09	3.236	2.31

4.4 Numerical Results

The proposed method was applied to model nonlinear electronic-photonic circuit/components to form time-domain models in three examples that are presented in this section. Results from these examples show significant improvement in computation time compared to existing techniques.

4.4.1 Physics-Based CMOS Driver

For the first example a four-stage CMOS driver including eight transistors connected to each other is considered (Transistors are equal sized using 1 μm technology). A schematic of this driver is shown in Figure 4.4.

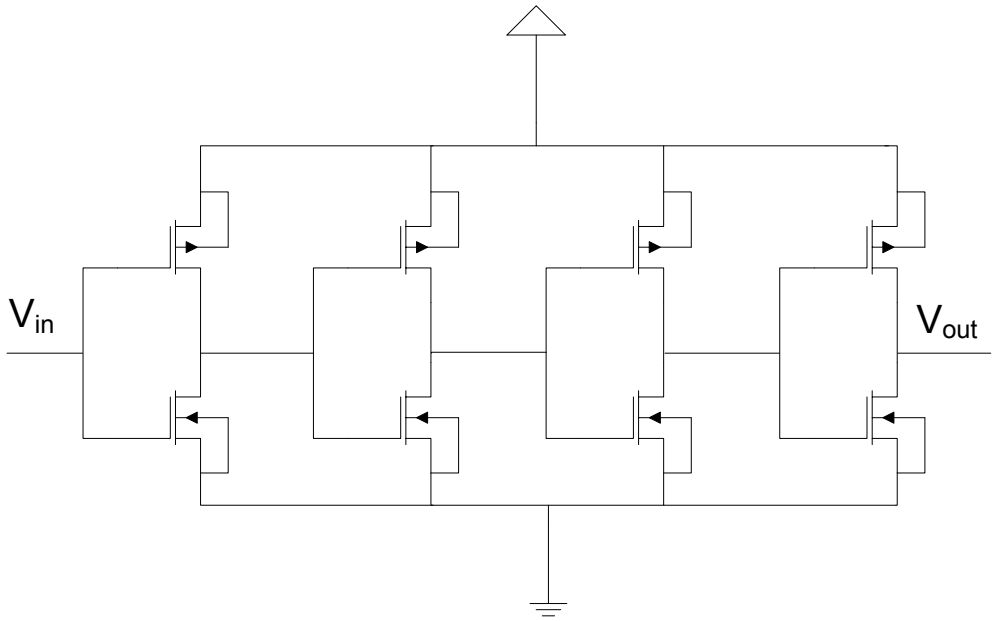


Figure 4.4: A 4-stage CMOS driver circuit used in Example 1.

This driver was initially modeled in physics-based simulator MINIMOS-NT [126] to perform transient simulation. Results from this simulation are presented in Figure 4.5 showing the input voltage waveform provided to the driver at V_{in} and the voltage waveform at the output of the driver. MINIMOS-NT is a physics-based simulator and, as mentioned previously, simulations using this software calls for time-consuming computations. This chapter addresses this issue by modeling CMOS

drivers using the ASSDNN technique (and also compared with the SSDNN technique) to form time-domain models that can be simulated along with other optical/electrical components.

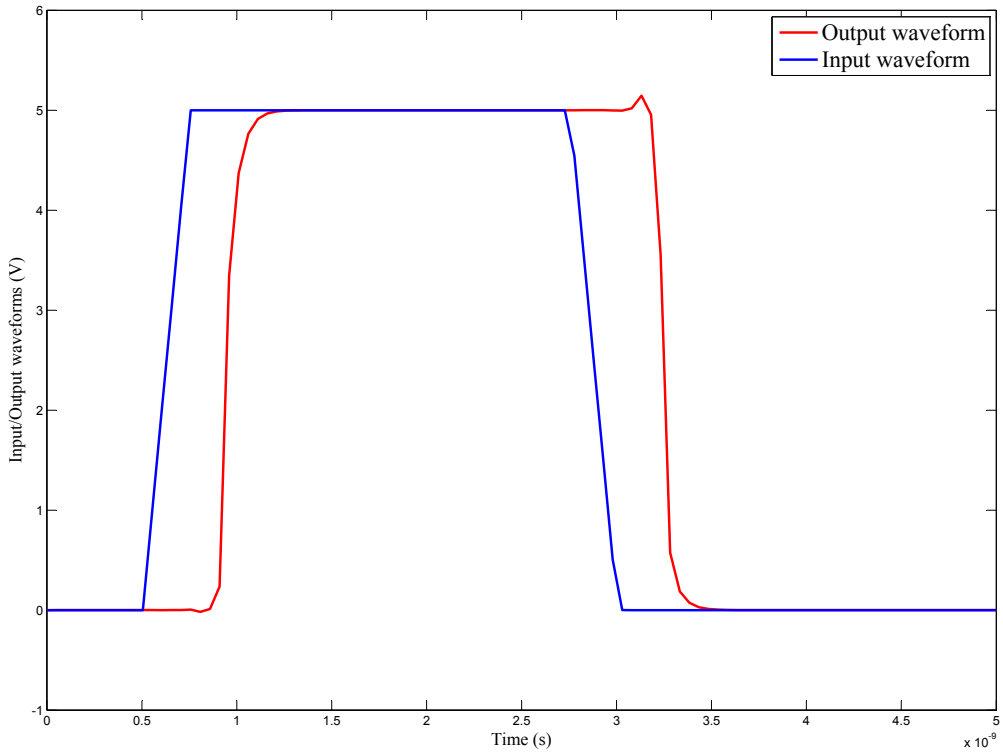


Figure 4.5: Input and output waveforms of 4-stage CMOS driver obtained using MINIMOS-NT.

Before using the ASSDNN technique to fully model and simulate the example, sample models based on ASSDNN and SSDNN methods were generated in order to compare the performance of ASSDNN with SSDNN. These two models are not the final models and the purpose for generating them was to compare the capability of these two methods. The input and output of these models were the voltages present

at the input and output of the driver. These inputs and outputs correspond to $u(t)$ and $y(t)$ respectively as described in Section 4.2 and are shown in Figure 4.6 (d and d' are desired output and its derivative respectively).

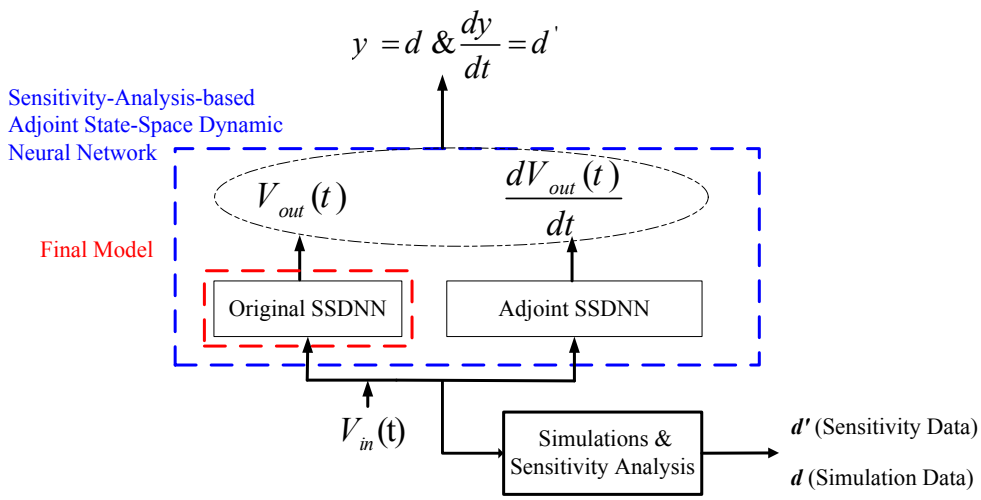


Figure 4.6: Structure of the model obtained by ASSDNN technique for the 4-stage CMOS driver.

Training data for this model was obtained from MINIMOS-NT simulations using which training was performed with 4 state variables and 10 hidden neurons (experimentally found) for both SSDNN and ASSDNN-based models. Input waveforms were obtained by changing rise/fall times (0.25ns, 0.5ns, 0.75ns) and amplitudes (4.5v, 5v, 5.5v). The input/output data were obtained without connecting load to the driver. The device can also be modeled when load is present but it should be trained for that. The training error of both models and their testing errors for two different waveforms that were not used in the training process are shown in Table 4.3. As seen from these results there is a clear advantage in using ASSDNN over

SSDNN to form nonlinear time-domain models. It is important to note that the superior capability of ASSDNN is due to the use of derivative information during training in comparison to SSDNN which does not use derivative information.

Table 4.3: Comparison between training and testing absolute errors of ASSDNN and SSDNN modeling of the 4-stage CMOS driver.

	Training error	Testing error for the 1_{st} test waveform	Testing error for the 2_{nd} test waveform
ASSDNN technique	25.4e-3	31.58e-3	100.55e-3
SSDNN technique	7.6e-4	73.68e-3	277.3e-3

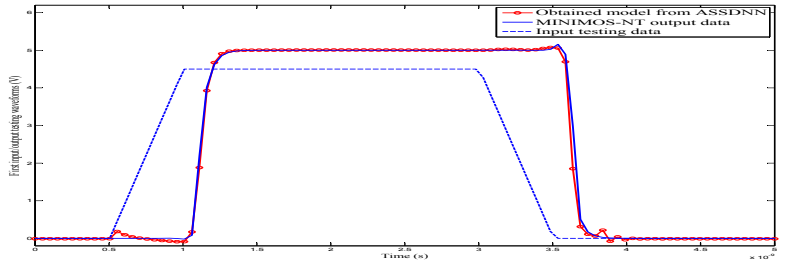
Further a model based on ASSDNN method was built to replace the CMOS driver in Figure 4.4 with 3 state variables and 18 hidden neurons using 6 training waveforms with. Data (including derivative information) was generated using MINIMOS-NT and used to train the model based on the ASSDNN technique. The transient model so obtained was used to simulate the electrical system. In addition, training of ASSDNN was performed using parallel computation and the results show a significant improvement in the time taken to generate the model. The time taken for simulation of the circuit with the model generated using ASSDNN and the simulation performed using MINIMOS-NT is shown in Table 4.4. As it can be seen from this table the time taken for simulating the circuit using the ASSDNN-based model

is much less than the time required to perform simulation using MINIMOS-NT. The results affirm the speed superiority of the model obtained by ASSDNN technique over the MINIMOS-NT model.

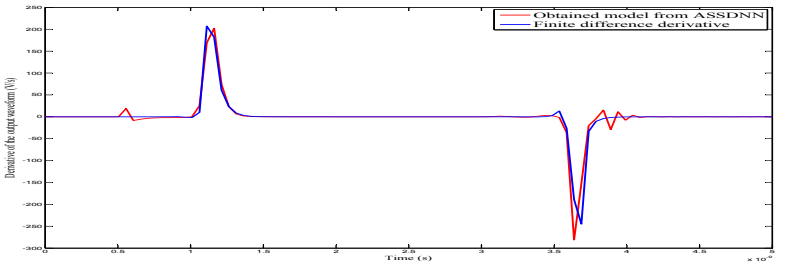
Table 4.4: Comparison between the CPU times of 1 waveform evaluation using the proposed ASSDNN and the physics-based MINIMOS-NT simulation tool for the 4-stage CMOS driver.

	CPU time for 1 waveform evaluation
ASSDNN	0.1387(s)
MINIMOS-NT	327.66(s)

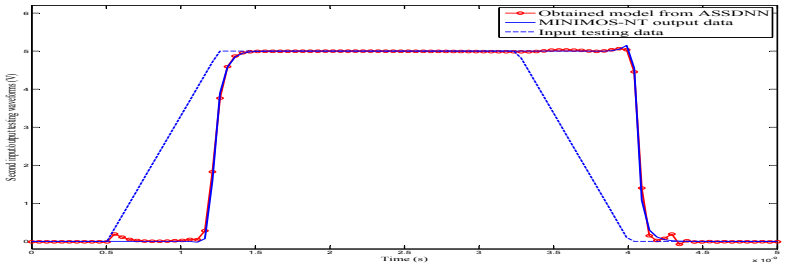
The final model obtained from ASSDNN was also validated with several independent testing waveforms. Figure 4.7 shows the comparison of the testing data and the response of the ASSDNN-based model for both the data and its derivative. As it can be seen in the figure the model obtained by ASSDNN technique matches the actual waveforms (from MINIMOS-NT) and their derivatives well with relatively small errors even though the testing waveforms were not included in the training data. Table 4.5 shows the testing error for each provided testing waveform.



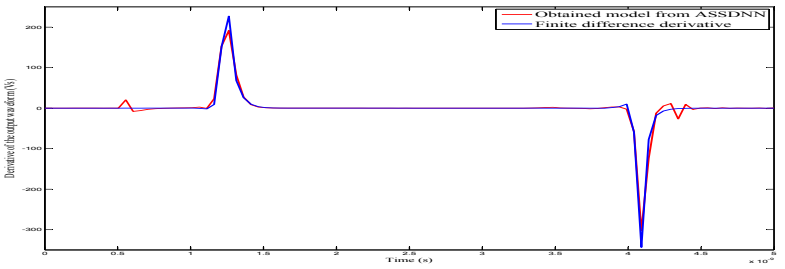
(a)



(b)



(c)



(d)

Figure 4.7: Testing waveforms for the validation of the full modeling of 4-stage CMOS driver based on ASSDNN technique. (a) and (b) The 1_{st} input/output testing waveforms and corresponding derivative, (c) and (d) The 2_{nd} input/output testing waveforms and corresponding derivative.

Table 4.5: Absolute testing errors of the provided test waveforms for the final obtained model of 4-stage CMOS driver using the ASSDNN technique.

	1_{st} test waveform	2_{nd} test waveform
Testing error	8.125e-3	1.924e-3

4.4.2 Optical Connection between 2 Cores of a Processor

For this example a microwave photonic link connecting two cores of a microprocessor is considered [127,128] as shown in Figure 4.8. Several signals are transmitted back and forth between both the cores of the microprocessor through the optical link between them. Electrical signals are converted to optical signals and are multiplexed onto the optical link through the use of ring-resonators. These signals are then demultiplexed and received by the other processor. The link is designed such that it mainly operates in the linear region; however as the number of signals multiplexed onto the link increases the intensity of the total optical signal present on the link increases pushing the link into the nonlinear region [123]. In this example, the link is considered when the intensity of the optical signal is large enough that nonlinearity sets in.

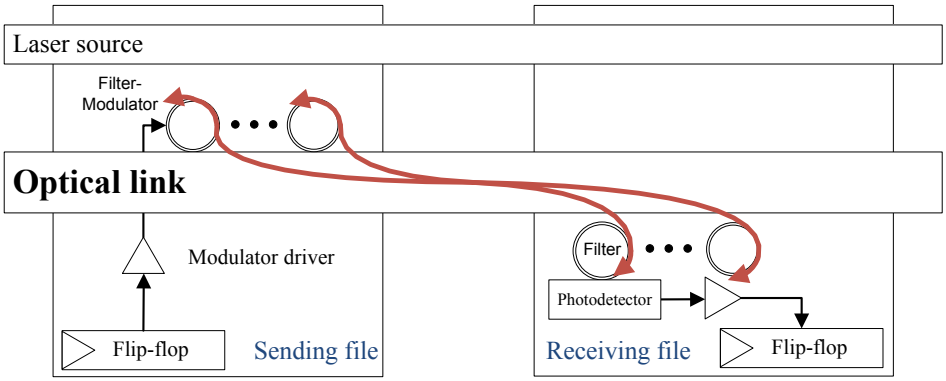


Figure 4.8: The schematic of the optical link between two cores.

In order to simulate this circuit the link exhibiting nonlinear behavior was initially modeled in OptiSPICE to perform transient simulation. Results from this simulation are presented in Figure 4.9. OptiSPICE models the link using SSF and, as previously mentioned, the transient engine of OptiSPICE [113] needs to perform convolutions in order to integrate the response of the link with the responses of other components present in the system. These convolutions are time-consuming and this chapter addresses this issue by modeling the link using the ASSDNN technique (and also compared with the SSDNN technique) to form models that can be simulated along with other components without the need for convolutions.

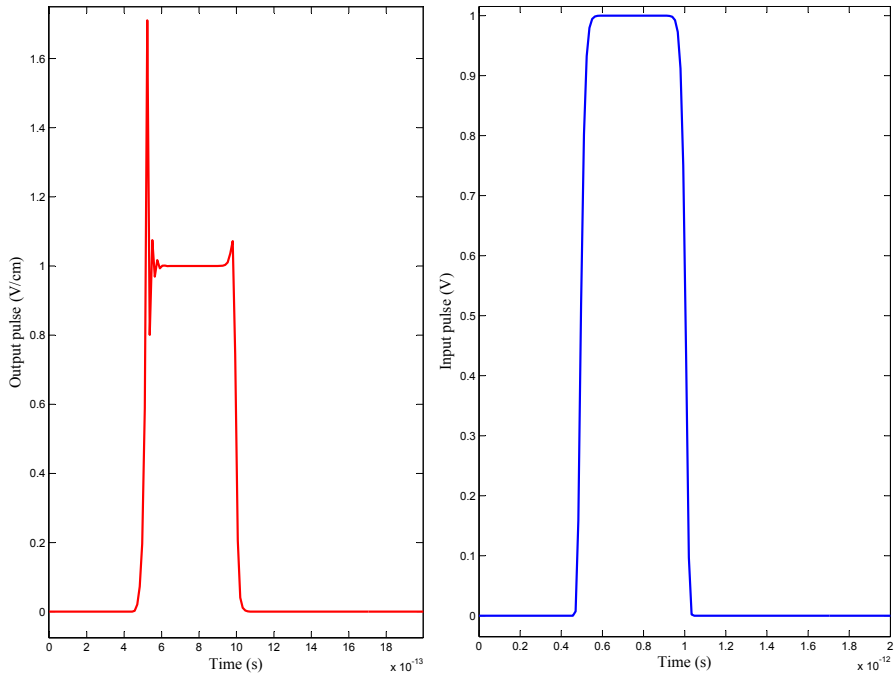


Figure 4.9: Input and output waveforms of the optical micro link between two cores obtained using OptiSPICE.

Similar to the previous example, in order to compare ASSDNN and SSDNN techniques, first, two sample models based on ASSDNN and SSDNN methods were created before going through full model generation. The input and output of these models, corresponding to $u(t)$ and $y(t)$ respectively (as described in Section 4.2), are the magnitude of the complex envelope of the electrical field present at the input and output of the optical link and are shown in Figure 4.10.

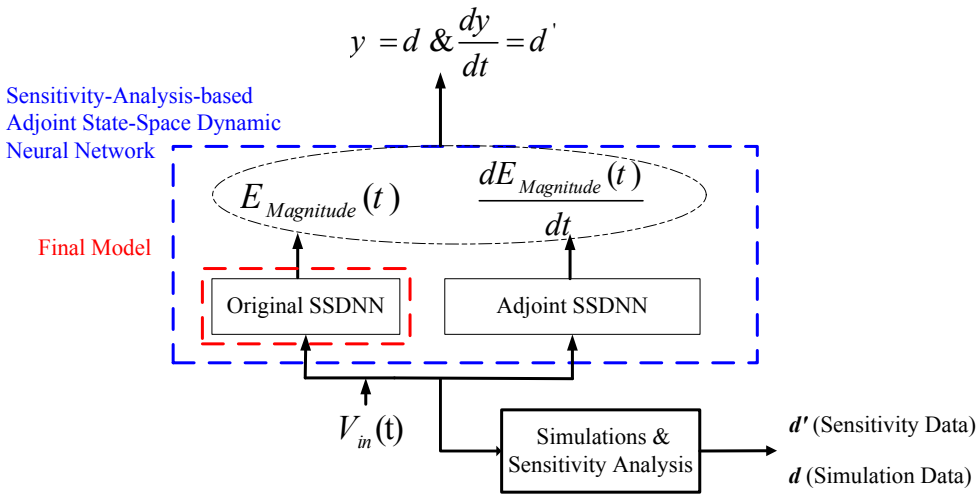


Figure 4.10: Structure of the model obtained by ASSDNN technique for the optical micro link between two cores.

The sample ASSDNN and SSDNN models used for training both had 4 state variables and 9 hidden neurons and the training data was acquired from OptiSPICE. Table 4.6 shows the training and testing errors of both models for two test waveforms that were not present in the training data. These results reaffirm the advantage of using ASSDNN over SSDNN to create nonlinear time-domain models due to the use of derivative information by ASSDNN during the training process.

Finally a model based on ASSDNN technique, with 4 state variables and 9 hidden neurons (experimentally determined) using 3 training waveforms, was created to represent the link in Figure 4.8. As there is an additional output demonstrating the derivative of the output signal associated with ASSDNN-based model, the corresponding training data was also included for the purpose of training. In this example, training data was generated using OptiSPICE and the obtained model

Table 4.6: Comparison between training and testing absolute errors of the models obtained by the proposed ASSDNN and the SSDNN methods for the optical micro link between two cores.

	Training error	Testing error for the 1_{st} test waveform	Testing error for the 2_{nd} test waveform
ASSDNN technique	7.69e-3	83.38e-3	80.33e-3
SSDNN technique	0.0021	117.64e-3	163.5e-3

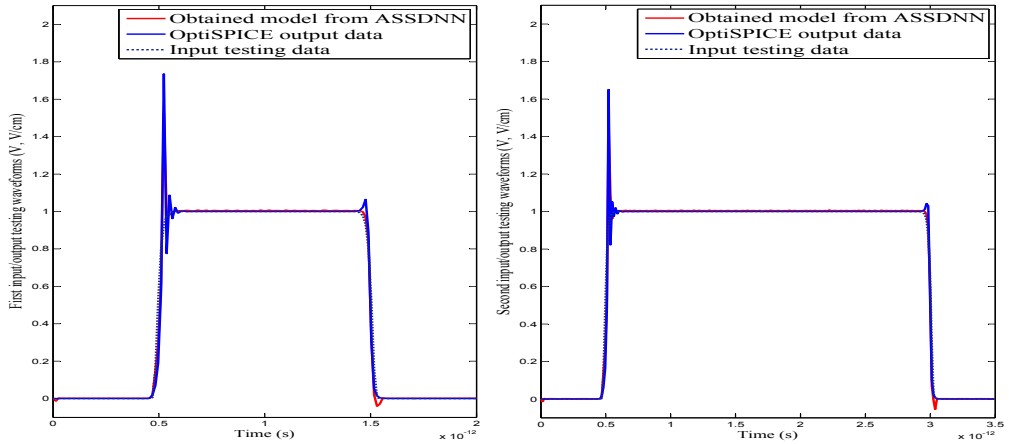
was used to perform simulations along with other components. Also, parallel computation was used to train and create the ASSDNN-based model which made the training process significantly faster. This speedup was obtained despite the fact that simulation using the ASSDNN-based model was performed using MATLAB whereas the simulation using OptiSPICE benefits from the framework being developed in the C programming language. Table 4.7 shows the comparison of the time taken to perform simulation by the obtained ASSDNN-based model and the OptiSPICE model. The results verifies the superiority of the model obtained by ASSDNN technique over the OptiSPICE model.

The full model obtained using ASSDNN method was also validated using several testing waveforms that were not included in the training data. A comparison of the testing data and the response of the ASSDNN-based model is demonstrated in

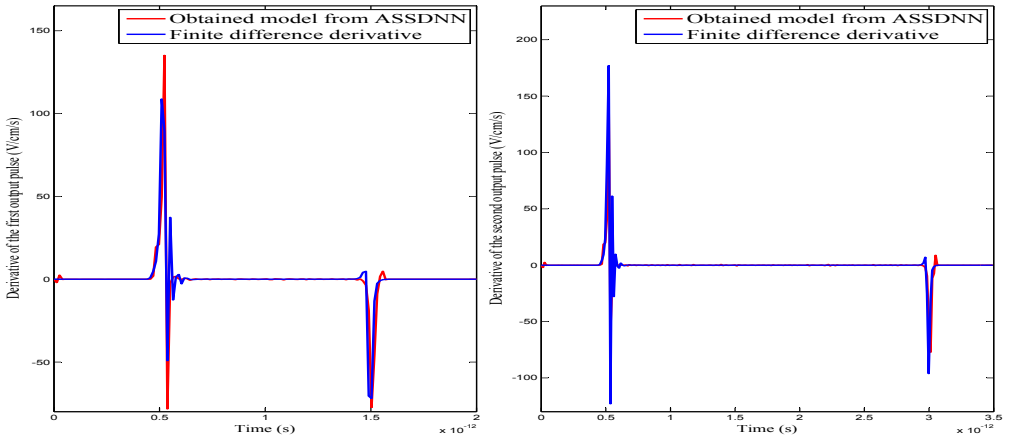
Table 4.7: Comparison between the evaluation time of models obtained by the proposed ASSDNN and the OptiSPICE simulation tool for the optical micro link between two cores.

	Evaluation time of the 1_{st} test waveform (512 bits)	Evaluation time of the 2_{nd} test waveform (1024 bits)
ASSDNN	12.41(s)	32.9(s)
OptiSPICE	94.25(s)	193.06(s)

Figure 4.11 which shows the accuracy of the obtained model. Also, the testing error for each test waveform is shown in Table 4.8.



(a)



(b)

Figure 4.11: Testing waveforms for the validation of the full modeling of the optical micro link between two cores based on ASSDNN technique. (a) The 1_{st} and 2_{nd} input/output testing waveforms, (b) The derivatives of 1_{st} and 2_{nd} testing waveforms.

4.4.3 Nonlinear Microring-Resonator

A nonlinear ring-resonator [129] was considered in this example and modeled using ASSDN and SSDNN. In OptiSPICE, the nonlinear ring-resonator was modeled using

Table 4.8: Absolute testing errors of the provided test waveforms for the final obtained model of the optical micro link between two cores using the ASSDNN technique.

	1_{st} test waveform	2_{nd} test waveform
Testing error	0.0097	0.001

couplers and linear and nonlinear waveguides. Figure 4.12 shows the schematic of a nonlinear ring-resonator.

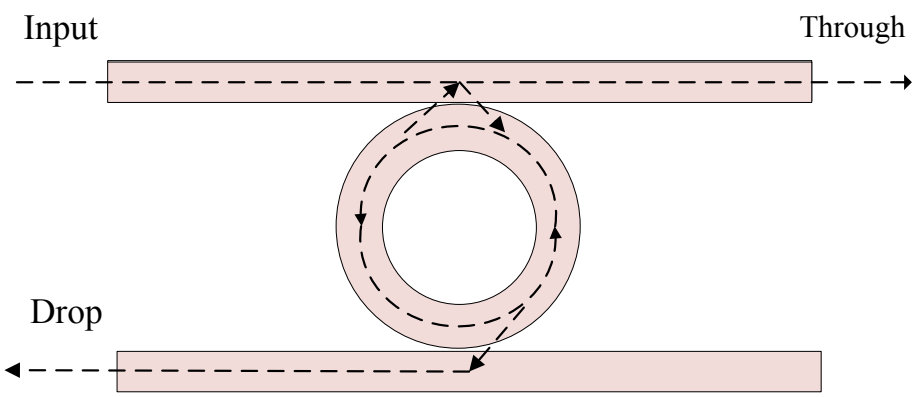


Figure 4.12: The schematic of a nonlinear ring-resonator.

In order to simulate this circuit the nonlinear ring was initially modeled in OptiSPICE to perform transient simulation. Results from this simulation are presented in Figure 4.13.

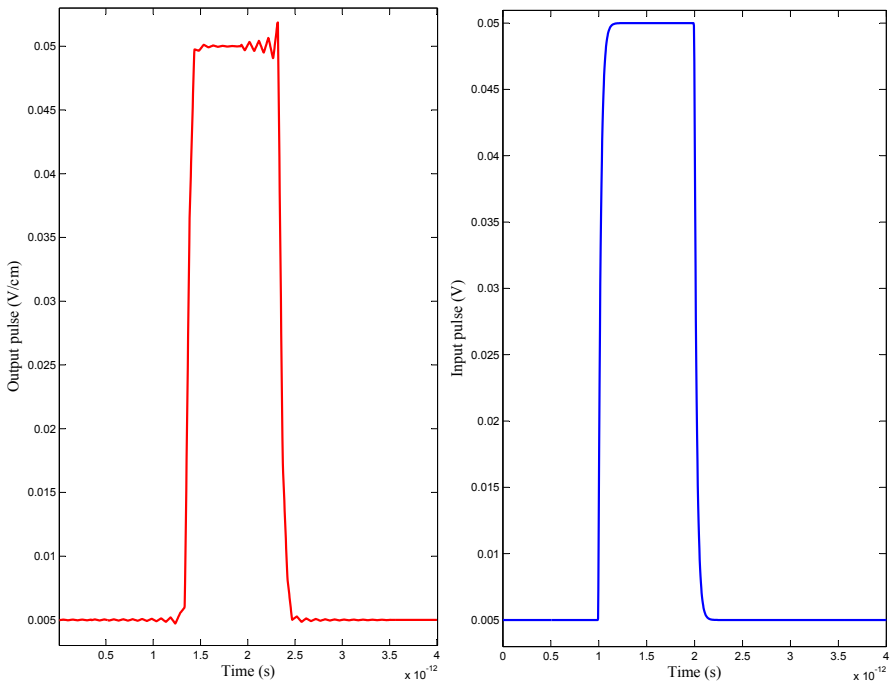


Figure 4.13: Input and output waveforms of the nonlinear microring-resonator obtained using OptiSPICE.

Similar to the previous examples, before going through the creation of the full model, two sample ASSDNN and SSDNN models are generated in order to compare the two techniques. Also, $u(t)$ and $y(t)$ (as explained in Section 4.2), the magnitude of the complex envelope of the electrical field present at the input and output of the ring, are the input and output of these models respectively.

Training data for this model was obtained from OptiSPICE simulations using which training was performed with 4 state variables and 9 hidden neurons for both SSDNN and ASSDNN-based models. Table 4.9 shows the training error of both models and also their testing errors using two different waveforms that were not used

in the training procedure. These obtained results again demonstrate how the use of derivative information during training process makes the ASSDNN technique much more capable than SSDNN to create time-domain models for nonlinear components.

Table 4.9: Comparison between training and testing absolute errors of the models obtained by the proposed ASSDNN and the SSDNN methods for the nonlinear ring-resonator.

	Training error	Testing error for the 1_{st} test waveform	Testing error for the 2_{nd} test waveform
ASSDNN technique	0.0104	0.055	1.38
SSDNN technique	0.0029	0.561	9.88

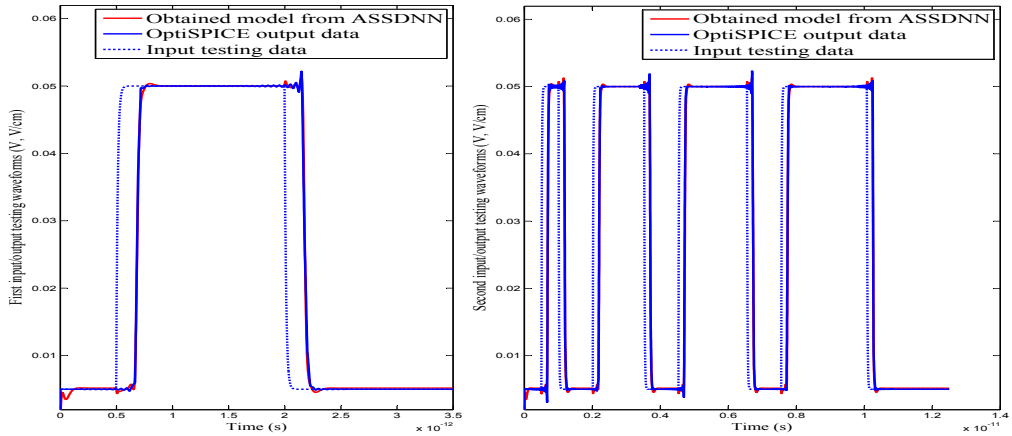
Eventually in order to replace the ring-resonator in Figure 4.12, a model based on ASSDNN technique with 4 state variables and 9 hidden neurons (experimentally obtained) using 3 training waveforms was generated. Similar to the previous example, training data was generated using OptiSPICE and the data corresponding to the derivative of the output signal was also provided for the purpose of training. Due to the use of parallel computation in the training process, model development was performed remarkably faster. A comparison between the simulation time of the model created by ASSDNN and the model in OptiSPICE is demonstrated in Table

4.10 for two long sample waveforms (256 and 512 bits). The results again confirm the efficiency of the ASSDNN-based model over the OptiSPICE model.

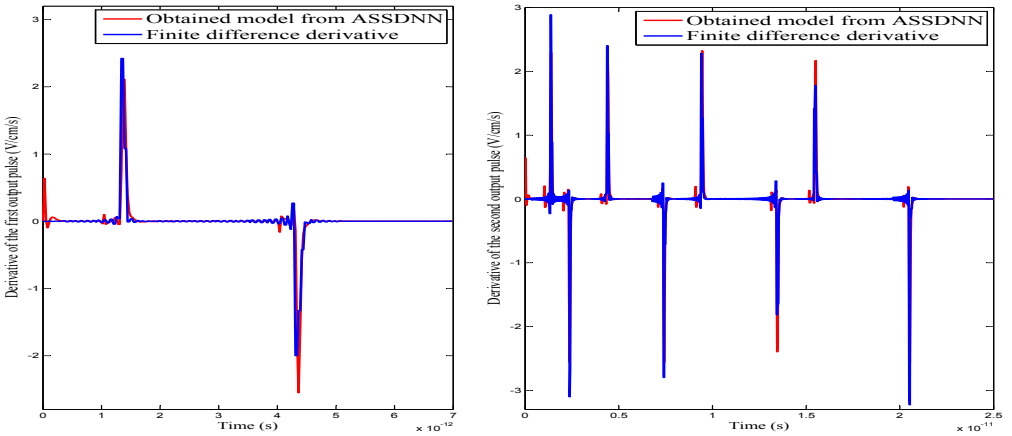
Table 4.10: Comparison between the evaluation time of models obtained by the proposed ASSDNN and the OptiSPICE simulation tool for the nonlinear ring-resonator.

	Evaluation time of the 1_{st} test waveform (256 bits)	Evaluation time of the 2_{nd} test waveform (512 bits)
ASSDNN	4.82(s)	25.75(s)
OptiSPICE	98.09(s)	284.97(s)

Further the full model for nonlinear ring-resonator obtained using ASSDNN method was also validated using several testing waveforms that were not included in the training data. Figure 4.14 exhibits the accuracy of the obtained model based on the proposed technique for the provided testing data. Also, Table 4.11 shows the testing errors for each test waveform.



(a)



(b)

Figure 4.14: Testing waveforms for the validation of the full modeling of the nonlinear ring resonator based on ASSDNN technique. (a) The 1_{st} and 2_{nd} input/output testing waveforms, (b) The derivatives of 1_{st} and 2_{nd} testing waveforms.

4.4.4 3-stage Inverting Buffer

In this example the transient modeling of a commercial IC package, namely inverting buffer 74LVC04A from NXP Semiconductors, is considered. For this component an

Table 4.11: Absolute testing errors of the provided test waveforms for the final obtained model of the nonlinear ring-resonator using the ASSDNN technique.

	1_{st} test waveform	2_{nd} test waveform
Testing error	0.0024	0.018

IBIS model as well as a detailed transistor-level model are readily available [130]. The IBIS model of this component is relatively fast but less accurate whereas the transistor-level model is relatively slow but more accurate. The schematic of this commercial device is shown in Figure 4.15.

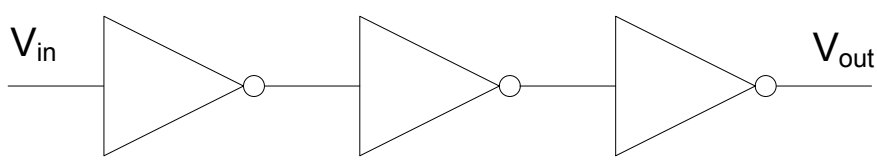
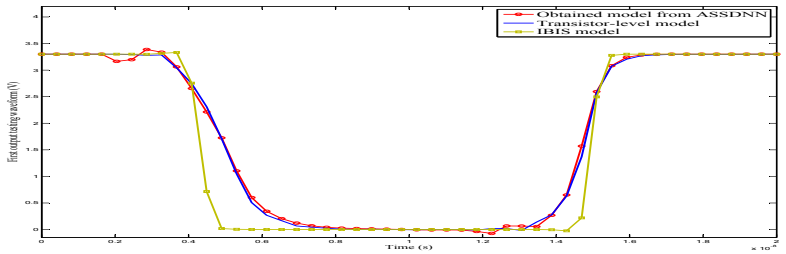


Figure 4.15: Schematic of NXP's 74LVC04A device based on its datasheet.

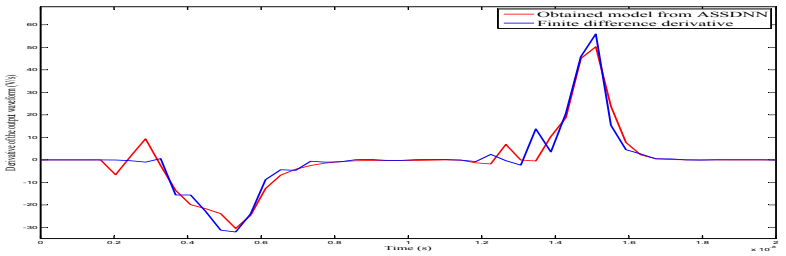
For fully modeling and simulating the 74LVC04A device, an ASSDNN-based model was built to replace the component in Figure 4.15 with 2 state variables and 10 hidden neurons (experimentally found) using 4 training waveforms. Input waveforms were obtained by changing rise/fall times (1.5ns, 1.75ns, 2ns) and amplitudes (3v, 3.3v, 3.6v). The inputs and outputs of this model correspond to $u(t)$ and $y(t)$ (the voltages at both ends of the buffer) respectively as described in Section 4.2.

The structure of this ASSDNN-based model is similar to Figure 4.6 for modeling the CMOS driver in the first example. Data for training this model was obtained from HSPICE simulations of the transistor-level model provided by NXP. Furthermore, the training process was executed using parallel computation and the time taken for generating the model was significantly improved.

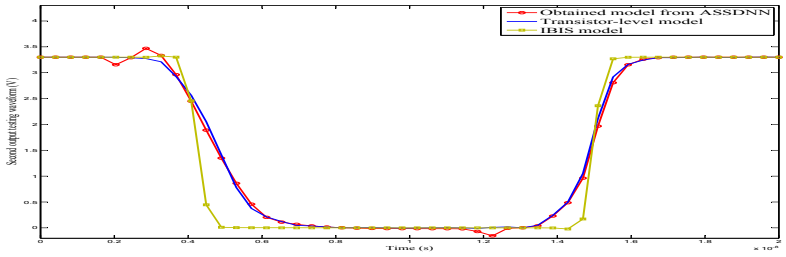
The final obtained ASSDNN-based model was also validated with several independent testing waveforms which were not used in the training procedure. Figure 4.16 shows the comparison of the response of the proposed ASSDNN-based model with IBIS and transistor-level models provided by NXP for these testing waveforms. Table 4.12 also demonstrates the comparison of the CPU time and accuracy of the proposed model with other aforementioned models. Note that the absolute errors in Table 4.12 were calculated relative to the transistor-level model and as such the error corresponding to the transistor-level model in Table 4.12 is zero. As can be seen from Figure 4.16 and Table 4.12 the ASSDNN-based model provides the best overall efficiency being faster than the transistor-level model and more accurate than the IBIS model while having a speed-up compared to the IBIS model. This demonstrates that ASSDNN-based models deliver both efficiency and accuracy which makes this technique the method of choice for modeling in VLSI/electronic design. Further it can be seen from Figure 4.16 that the obtained ASSDNN-based model matches the sensitivities with desirable accuracy.



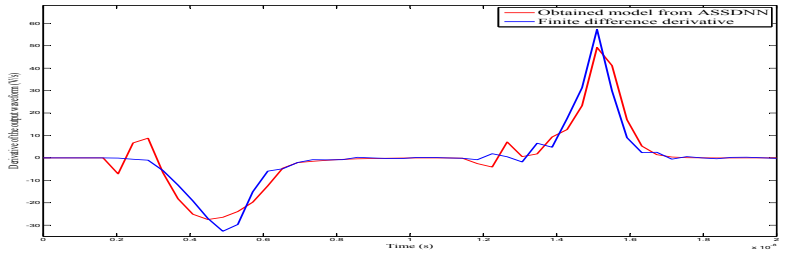
(a)



(b)



(c)



(d)

Figure 4.16: Testing waveforms for the validation and comparison of the ASSDNN-based model with the IBIS and transistor-level models for 74LVC04A inverting buffer. (a) and (b) The 1_{st} input/output testing waveforms and corresponding derivative, (c) and (d) The 2_{nd} input/output testing waveforms and corresponding derivative.

Table 4.12: Comparison of CPU time and accuracy for the proposed ASSDNN-based model and IBIS model of NXP's 74LVC04A device for sample test waveforms.

	ASSDNN-based Model	IBIS Model	Transistor-level Model
Speed-up ratio for 200-bit long test waveform	15.96	11.27	1 (reference for comparison)
Absolute test error for a waveform that was not used in training	2.15e-3	69.7e-3	0.0 (reference for comparison)

4.5 Summary and Conclusion

In this chapter a novel technique to model nonlinear circuits was presented. Building upon state-space dynamic neural networks this technique uses sensitivity (derivative) information during training of the dynamic neural network to generate time-domain models with greater accuracy for the same training data. Numerical comparisons demonstrating the efficiency obtained in training were presented in this chapter. Further speed-up resulting from faster training due to use of derivative information and parallellization was also demonstrated. Simulations using models obtained from training nonlinear microwave electronic-photonic circuits and components using the proposed technique were compared with simulations performed using the optical and electrical simulation tools, OptiSPICE and MINIMOS-NT, and a

significant speed-up was observed. This speed-up was obtained despite the fact that the models generated using the proposed technique were simulated using MATLAB whereas simulations using OptiSPICE have the advantage that OptiSPICE which is a commercial simulation package was implemented using the C programming language. It is naturally expected that if evaluation and simulation of ASSDNN-based models is performed in C, a much greater speed-up would be obtained.

Chapter 5

Conclusions and Future Research

5.1 Conclusions

In this thesis, two new methods for modeling VLSI/Electronic, photonic and microwave components and systems are presented. Both techniques adds the sensitivity information to the outputs of the conventional training methods resulting in the generation of models with more accuracy for similar training data.

The first technique, sensitivity-analysis-based artificial neural network (SAANN), is an advance over conventional static multilayer perceptron (MLP) which adds sensitivity information to training process resulting in less training data required for training. The obtained model provides additional sensitivity outputs with respect to all the inputs.

The second proposed method, adjoint state-space dynamic neural network (ASS-DNN), is an advance over conventional state-space dynamic neural network (SS-DNN) training method. It adds the time derivative information to the training process resluting in lee time-steps required for training. It also provided additional

derivative outputs with respect to time. In addition, ASSDNN was developed so that it can take the advantage of parallel computation resulting further speedup. Several optical/electrical examples are provided to demonstrate the accuracy of the proposed techniques.

Also, comparisons have been made between the training process of the proposed SAANN and ASSDNN methods and the conventional training methods, MLP and SSDNN. Further, the simulations have been performed using optical and electrical simulation tools, OptiSPICE and MINIMOS-NT, and EM simulation tool (CST) and the results have been compared with simulations using the proposed techniques. Comparisons demonstrate the advantage and superiority of the proposed methods over both conventional training techniques and evaluations using simulations tools in addition to providing all sensitivity information that are not available in simulation tools. Noteworthy to mention that simulations using the proposed ASSDNN technique were performed in MATLAB whereas simulations using OptiSPICE have the advantage of being implemented using the C programming language. It is likely that if ASSDNN simulations are also performed in C there would be an even greater speedup compared to OptiSPICE.

5.2 Future Research

Given below are some of the future directions that can be taken to continue the work that has been initiated in this thesis to develop neural networks using sensitivity information:

- Development of sensitivity analysis-based techniques for discrete-time ANN

techniques such as recurrent neural networks (RNN), as a fundamental type of ANN structure, in order to make RNN-based modeling techniques require less training data consequently resulting in more efficient model development.

- In addition, this modeling technique can be modified to be used with parallel computation which could potentially increase the speedup significantly.
- The robustness of this method can be characterized against the presence of noise in the training data.
- As MATLAB uses the finite difference for calculating the Hessian (second derivative) of the error function, if the Hessian is mathematically calculated and provided to the optimization toolbox, it can speedup the optimization (training) process. This work can be done for all SSDNN, ASSDNN, RNN or the adjoint RNN techniques. It should be noted that Hessian for the adjoint method requires third order derivatives which is mathematically hard to find and expensive.
- Study the possibility of parallelization of time-domain training using GPU.

References

[1] P. M. Watson, and K. C. Gupta, "EM-ANN Models for Microstrip Vias and Interconnects in Multilayer Circuits," *IEEE Trans. Microwave Theory and Techniques,* Vol. 44, Dec. 1996, pp. 2495-2503.

[2] G. L. Creech, et al., "Artificial Neural Networks for Fast and Accurate EM-CAD of Microwave Circuits," *IEEE Trans. Microwave Theory and Techniques,* Vol. 45, May 1997, pp. 794-802.

[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. on Microwave Theory and Techniques,* Vol. 45, May 1997, pp. 712-723.

[4] A. H. Zaabab, Q. J. Zhang, and M. Nakhla, "A Neural Network Modeling Approach to Circuit Optimization and Statistical Design," *IEEE Trans. on Microwave Theory and Techniques,* Vol. 43, June 1995, pp. 1349-1358.

[5] S. N. Balakrishnan, and R. D. Weil, "Neurocontrol: A Literature Survey," *Mathematical and Computer Modeling,* Vol. 23, No. 1-2, January 1996, pp. 101-117.

[6] B. S. Cooper, "Selected Applications of Neural Networks in Telecommunication Systems," *Australian Telecommunication Research,* Vol. 28, No. 2, 1994, pp. 9-29.

[7] T. Alvager, T. J. Smith, and F. Vijai, "The Use of Artificial Neural Networks in Biomedical Technologies: An Introduction," *Biomedical Instrumentation and Technology,* Vol. 28, No. 4, Jul-Aug 1994, pp. 315-322.

[8] K. Goita, et al., "Literature Review of Artificial Neural Networks and Knowledge Based Systems for Image Analysis and Interpretation of Data in Remote Sensing," *Canadian Journal of Electrical and Computer Engineering,* Vol. 19, No. 2, April 1994, pp. 53-61.

[9] Y. G. Smetanin, "Neural Networks as Systems for Pattern Recognition: A Review," *Pattern Recognition and Image Analysis,* Vol. 5, No. 2, 1995, pp. 254-293.

[10] J. F. Jr. Nunmaker, and R. H. Sprague Jr., "Applications of Neural Networks in Manufacturing," *Proceedings of the Twenty-ninth Hawaii International Conference on System Sciences,* Vol. 2, 1996, pp. 447-453.

[11] Q. J. Zhang, and G. L. Creech (Guest Editors), International Journal of RF and Microwave Computer-Aided Engineering, *Special Issue on Applications of Artificial Neural Networks to RF and Microwave Design,* Vol. 9, NY: Wiley, 1999.

[12] M. Vai, and S. Prasad, "Automatic Impedance Matching with a Neural Network", *IEEE Microwave and Guided Wave Letters,* Vol. 3, No. 10, Oct. 1993, pp. 353-354.

[13] T. Horng, C. Wang, and N. G. Alexopoulos, "Microstrip Circuit Design Using Neural Networks," *MTT-S Int. Microwave Symp. Dig.,* 1993, pp. 413-416.

[14] A. H. Zaabab, Q. J. Zhang, and M. Nakhla, "Analysis and Optimization of Microwave Circuits and Devices Using Neural Network Models," *MTT-S Int. Microwave Symp. Dig.,* 1994, pp. 393-396.

[15] V. B. Litovski, et. al., "MOS Transistor Modeling Using Neural Network," *Electronics Letters,* Vol. 28, No. 18, 1992, pp. 1766-1768.

[16] F. Gunes, F. Gurgen, and H. Torpi, "Signal-Noise Neural Network Model for Active Microwave Devices," *IEE Proc.-Circuits, Devices, Syst.,* Vol. 143, No. 1, Feb. 1996, pp. 1-8.

[17] K. Shirakawa, et. al., "A Large-Signal Characterization of an HEMT Using a Multilayered Neural Network," *IEEE Trans. on Microwave Theory and Techniques,* Vol. 45, No. 9, Sept. 1997, pp. 1630-1633.

[18] P. M. Watson, C. Cho, and K. C. Gupta, "EM-ANN Model Synthesis of Physical Dimensions for Multilayer Asymmetric Coupled Transmission Line Structures," *International Journal of RF and Microwave Computer-Aided Engineering,* Vol. 9, No. 3, 1999, pp. 175-186.

[19] P. M. Watson, K. C. Gupta, and R. L. Mahajan, "Development of Knowledge Based Artificial Neural Network Models for Microwave Components," *IEEE MTT-S Int. Microwave Symp.,* 1998, Digest, pp. 9-12.

[20] Q. J. Zhang, et al., "Ultra Fast Neural Models for Analysis of Electro/Opto Interconnects," *IEEE Electronic Components and Technology Conf.,* San Jose, CA, May 1997, pp. 1134-1137.

[21] Q. J. Zhang, F. Wang, and V. Devabhaktuni, "Neural Network Structures for RF and Microwave Applications," *IEEE AP-S Antennas and Propagations International Symp.,* (Orlando, FL), July 1999, pp. 2576-2579.

[22] F. Wang, et al., "Neural Network Structures and Training Algorithms for Microwave Applications," *International Journal of RF and Microwave CAE,* Special Issue on Applications of Artificial Neural Networks to RF and Microwave Design, Vol. 9, 1999, pp. 216-240.

[23] V. Devabhaktuni, C. Xi, F. Wang, and Q. J. Zhang, "Robust Training of Microwave Neural Models," *IEEE MTT-S International Microwave Symp.,* (Anaheim, CA), June 1999, Digest, pp. 145-148.

[24] N. Dong and J. Roychowdhury, "Automated nonlinear macromodelling of output buffers for high-speed digital applications", 42th Proceedings Design Automation Conference, 2005, pp. 51-56.

[25] F. Wang, and Q. J. Zhang, "Incorporating Functional Knowledge into Neural Networks," *IEEE Int. Conf. Neural Networks,* (Houston, TX), June 1997, pp. 266-269.

[26] F. Wang, and Q. J. Zhang, "Knowledge-Based Neural Models for Microwave Design," *IEEE Trans. on Microwave Theory and Techniques,* Vol. 45, Dec. 1997, pp. 1349-1358.

[27] D. Wu, et al., "Accurate Numerical Modeling of Microstrip Junctions and Discontinuities," *Int. J. Microwave mm-Wave Computer-Aided Eng.,* Vol. 1, No. 1, 1991, pp. 48-58.

[28] J. Aweya, Q. J. Zhang, and D. Montuno, "Neural Sensitivity Methods for the Optimization of Queueing Systems," *1998 World MultiConference on Systematics, Cybernetics and Infomatics,* Orlando, Florida, July 1998 (invited), pp. 638-645.

[29] F. Scarselli, and A. C. Tsoi, "Universal Approximation using Feedforward Neural Networks: A Survey of Some Existing Methods, and Some New Results," *Neural Networks,* Vol. 11, 1998, pp. 15-37.

[30] G. Cybenko, "Approximation by Superpositions of a Sigmoidal Function," *Math. Control Signals Systems,* Vol. 2, 1989, pp. 303-314.

[31] K. Hornik, M. Stinchcombe, and H. White, "Multilayer Feedforward Networks are Universal Approximators," *Neural Networks,* Vol. 2, 1989, pp. 359-366.

[32] 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, 1997, pp. 630-645.

[33] R. Reed, "Pruning AlgorithmsA Survey," *IEEE Trans. Neural Networks,* Vol. 4, Sept. 1993, pp. 740-747.

[34] A. Krzyzak, and T. Linder, "Radial Basis Function Networks and Complexity Regularization in Function Learning," *IEEE Trans. Neural Networks,* Vol. 9, 1998, pp. 247-256.

[35] J. de Villiers, and E. Barnard, "Backpropagation Neural Nets with One and Two Hidden Layers," *IEEE Trans. Neural Networks,* Vol. 4, 1992, pp. 136-141.

[36] S. Tamura, and M. Tateishi, "Capabilities of a Four-Layered Feedforward Neural Network: Four Layer Versus Three," *IEEE Trans. Neural Networks,* Vol. 8, 1997, pp. 251-255.

[37] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning Internal Representations by Error Propagation," in *Parallel Distributed Processing,* Vol. 1, D.E. Rumelhart and J. L. McClelland, Editors, Cambridge, MA: MIT Press, 1986, pp. 318-362.

[38] J. A. Garcia, et al., "Modeling MESFET's and HEMT's Intermodulation Distortion Behavior using a Generalized Radial Basis Function Network," *Int. Journal of RF and Microwave CAE,* Special Issue on Applications of ANN to RF and Microwave Design, Vol. 9, 1999, pp. 261-276.

[39] I. S. Stievano, I. A. Maio, and F. G. Canavero, "Parametric Macromodels of Digital I/O Ports" , *IEEE Transactions on Advanced Packaging,* vol. 25, no. 2, pp. 225-264, May, 2002.

[40] I. S. Stievano, I. A Maio, and F.G. Canavero, "Mpilog macromodeling via parametric identification of logic gates," *IEEE Trans. Adv. Packag. ,* vol. 27, no. 1, pp. 15-23, Feb. 2004.

[41] J. Aweya, Q. J. Zhang, and D. Montuno, "A Direct Adaptive Neural Controller for Flow Control in Computer Networks," *IEEE Int. Conf. Neural Networks,* Anchorage, Alaska, May 1998, pp. 140-145.

[42] J. Aweya, Q. J. Zhang, and D. Montuno, "Modelling and Control of Dynamic Queues in Computer Networks using Neural Networks," *IASTED Int. Conf. Intelligent Syst. Control,* Halifax, Canada, June 1998, pp. 144-151.

[43] L. H. Tsoukalas, and R. E. Uhrig, *Fuzzy and Neural Approaches in Engineering,* NY: Wiley-Interscience, 1997.

[44] J. A. Freeman, and D. M. Skapura, *Neural Networks: Algorithms, Applications and Programming Techniques,* Reading, MA: Addision-Wesley, 1992.

[45] J. J. Xu, M. C. E. Yagoub, R. Ding, and Q. J. Zhang, "Neural-based dynamic modeling of nonlinear microwave circuits," *IEEE Trans. Microw. Theory Tech. ,* vol. 50, no. 12, pp. 2769-2780, Dec. 2002.

[46] R. Battiti, "Accelerated Backpropagation Learning: Two Optimization Methods," *Complex Systems,* Vol. 3, 1989, pp. 331-342.

[47] M. Arisawa, and J. Watada, "Enhanced Backpropagation Learning and its Application to Business Evaluation," In *Proc. IEEE Intl. Conf. Neural Networks,* Vol. I, Orlando, Florida, July 1994, pp. 155-160.

[48] X. H. Yu, G. A. Chen, and S. X. Cheng, "Dynamic Learning Rate Optimization of the Backpropagation Algorithm," *IEEE Trans. Neural Networks,* Vol. 6, May 1995, pp. 669-677.

[49] K. Ochiai, N. Toda, and S. Usui, "Kick-out Learning Algorithm to Reduce the Oscillation of Weights," *Neural Networks,* Vol. 7, 1994, pp. 797-807.

[50] D. B. Parker, "Optimal Algorithms for Adaptive Networks: Second Order Backpropagation, Second Order Direct Propagation and Second Order Hebbian Learning," In *Proc. IEEE First Intl. Conf. Neural Networks,* Vol. II, San Diego, California, 1987, pp. 593-600.

[51] R. Battiti, "online First- and Second-Order Methods for Learning: Between Steepest Descent and Newton's Method," *Neural Computation,* vol. 4, pp. 141-166, Feb. 1992.

[52] W. H. Press, et al., *Numerical Recipes: The Art of Scientific Computing,* Cambridge, UK: Cambridge University Press, 1992.

[53] R. Fletcher, and C. M. Reeves, "Function Minimization by Conjugate Gradients," *Computer Journal,* Vol. 6, 1964, pp. 149-154.

[54] E. Polak, and G. Ribiere, "Note sur la Convergence de Mthode de Directions Conjugues," *Revue Francaise Informat. Rechercher Operationnelle,* Vol. 16, 1969, pp. 35-43.

[55] Q. J. Zhang and K. C. Gupta, *Neural Networks for RF and Microwave Design.* Norwood, MA: Artech House, 2000.

[56] T. R. Cuthbert, Jr., "Quasi-Newton Methods and Constraints," In *Optimization using Personal Computers,* NY: John Wiley and Sons, 1987, pp. 233-314.

[57] W. C. Davidon, *Variable Metric Method for Minimization,* Research and Development Report ANL-5990, U.S. Atomic Energy Commission, Argonne National Laboratories, 1959.

[58] C. G. Broyden, "Quasi-Newton Methods and their Application to Function Minimization," *Math. Comp.* Vol. 21, 1967, pp. 368-381.

[59] K. R. Nakano, "Partial BFGS Update and Efficient Step-Length Calculation for Three-Layer Neural Networks," *Neural Computation,* Vol. 9, 1997, pp. 123-141.

[60] S. McLoone, and G. W. Irwin, "Fast Parallel Off-Line Training of Multilayer Perceptrons," *IEEE Trans. Neural Networks,* Vol. 8, May 1997, pp. 646-653.

[61] J. E. Rayas-Snchez, "EM-based optimization of microwave circuits using artificial neural networks: the state-of-the-art," *IEEE Trans. Microwave Theory Tech.,* vol. 52, no. 1, pp. 420-435, Jan. 2004.

[62] V. Rizzoli, A. Costanzo, D. Masotti, A. Lipparini, and F. Mastri, "Computer-aided optimization of nonlinear microwave circuits with the aid of electromagnetic simulation," *IEEE Trans. Microwave Theory Tech.,* vol. 52, no. 1, pp. 362-377, Jan. 2004.

[63] M. B. Steer, J.W. Bandler, and C. M. Snowden, "Computer-aided design of RF and microwave circuits and systems," *IEEE Trans. Microwave Theory Tech.,* vol. 50, no. 3, pp. 996-1005, Mar. 2002.

[64] P. Burrascano, S. Fiori, and M. Mongiardo, "A review of artificial neural networks applications in microwave computer-aided design," *Int. J. RF and Microw. CAE,* vol. 9, no. 3, pp. 158-174, May 1999.

[65] Q.J. Zhang, K. C. Gupta, and V. K. Devabhaktuni, "Artificial neural networks for RF and microwave designfrom theory to practice," *IEEE Trans. Microwave Theory Tech.,* vol. 51, no. 4, pp. 1339-1350, Apr. 2003.

[66] V.K. Devabhaktuni, B. Chattaraj, M. C. E. Yagoub, and Q.J. Zhang, "Advanced microwave modeling framework exploiting automatic model generation, knowledge neural networks, and space mapping," *IEEE Trans. Microwave Theory Tech.,* vol. 51, no. 7, pp. 1822-1833, July 2003.

[67] S. Koziel and J.W. Bandler, "A space-mapping approach to microwave device modeling exploiting fuzzy systems," *IEEE Trans. Microwave Theory Tech.,* vol. 55, no. 12, pp. 2539-2547, Dec. 2007.

[68] J. E. Rayas-Snchez and V. Gutirrez-Ayala, "EM-based monte carlo analysis and yield prediction of microwave circuits using linear-input neural-output space mapping," *IEEE Trans. Microwave Theory Tech.* vol. 54, no. 12, pp. 4528-4537, Dec. 2006.

[69] Y. Cao and G. Wang, "A wideband and scalable model of spiral inductors using space-mapping neural network," *IEEE Trans. Microwave Theory Tech.,* vol. 55, no. 12, pp. 2473-2480, Dec. 2007.

[70] Y. Cao, G. Wang, and Q. J. Zhang, "A new training approach for parametric modeling of microwave passive components using combined neural networks and transfer functions," *IEEE Trans. Microwave Theory Tech.,* vol. 57, no. 11, pp. 2727-2742, Nov. 2009.

[71] *CST MICROWAVE STUDIO(R)* (2010), CST AG, Bad Nauheimer Str. 19, D-64289 Darmstadt, Germany, 2010. http://www.cst.com.

[72] HFSS. Ansoft Corporation, Canonsburg, PA, USA, 2007. [Online]. Available: http://www.ansoft.com/products/hf/hfss/

[73] N. K. Nikolova, J. Zhu, D. Li, M. H. Bakr, and J. W. Bandler, "Sensitivity analysis of network parameters with electromagnetic frequency-domain simulators," *IEEE Trans. Microw. Theory Techn.,* vol. 54, no. 2, pp. 670681, Feb. 2006.

[74] Q. S. Cheng, J. W. Bandler, N. K. Nikolova, and S. Koziel, "Fast space mapping modeling with adjoint sensitivity," in *IEEE MTT-S Int. Microw. Symp. Dig.,* Baltimore, MD, USA, Jun. 2011.

[75] M. H. Bakr, N. K. Nikolova, and P. A. W. Basl, "Self-adjoint S-parameter sensitivities for lossless homogeneous TLM problems," *Int. J. Numer. Model.,* vol. 18, no. 6, pp. 441455, Nov. 2005.

[76] O. S. Ahmed, M. H. Bakr, X. Li, and T. Nomura, "A time-domain adjoint variable method for materials with dispersive constitutive parameters," *IEEE Trans. Microwave Theory Tech.,* vol. 60, no. 10, October 2012.

[77] N. Uchida, S. Nishiwaki, K. Izui, M. Yoshimura, T. Nomura, and K. Sato "Simultaneous shape and topology optimization for the design of patch antennas," *Proc. Antennas Propag.,* Mar. 2009, pp. 103107.

[78] M. H. Bakr, M. Ghassemi, and N. Sangary, "Bandwidth enhancement of narrow band antennas exploiting adjoint-based geometry evolution," *Proc. IEEE Int. Antennas Propag. Symp.,* Jul. 2011, pp. 29092911.

[79] A. Khalatpour, R. K. Amineh, Q. S. Cheng, M. H. Bakr, N. K. Nikolova, and J. W. Bandler, "Accelerating space mapping optimization with adjoint sensitivities," *IEEE Microw. Wireless Compon. Lett.,* vol. 21, no. 6, pp. 280282, Jun. 2011.

[80] O. Stan and E. Kamen, "A local linearized least squares algorithm for training feedforward neural networks," *IEEE Trans. Neural Netw.,* vol. 11, no. 2, pp. 487495, Mar. 2000.

[81] Y. Xu, K.-W. Wong, and C.-S. Leung, "Generalized RLS approach to the training of neural networks," *IEEE Trans. Neural Netw.,* vol. 17, no. 1, pp. 1934, Jan. 2006.

[82] K. C. Lee, "Application of neural network and its extension of derivative to scattering from a nonlinearly loaded antenna," *IEEE Trans. Antennas Propag.,* vol. 55, no. 3, pp. 990993, Mar. 2007.

[83] J. Xu, M. C. E. Yagoub, R. Ding, and Q. J. Zhang, "Exact adjoint sensitivity analysis for neural-based microwave modeling and design," *IEEE Trans. Microwave Theory Tech.,* vol. 51, no. 1, pp. 226-237, Jan. 2003.

[84] Q. J. Zhang, "NeuroModeler plus," Dept. Electron., Carleton Univ., Ottawa, ON, Canada, 2005.

[85] S. R. Schmidt and R. G. Launsby, "Understanding industrial designed experiments," Air Force Acad., Colorado Springs, CO, USA, 1992.

[86] L. Zhang, J. J. Xu, M. C. E. Yagoub, R. T. Ding, and Q. J. Zhang, "Efficient analytical formulation and sensitivity analysis of neuro-space mapping technique for nonlinear microwave device modeling," *IEEE Trans. Microw. Theory Techn.*, vol. 53, no. 9, pp. 2752-27767, Sep. 2005.

[87] Y. Fang, M. C. E. Yagoub, F. Wang, and Q. J. Zhang, "A new macromodeling approach for nonlinear microwave circuits based on recurrent neural networks," *IEEE Trans. Microw. Theory Techn.*, vol. 48, no. 12, pp. 2335-2344, Dec. 2000.

[88] J. Xu, M. C. E. Yagoub, R. Ding, and Q. J. Zhang, "Neural based dynamic modeling of nonlinear microwave circuits," *IEEE Trans. Microw. Theory Techn.*, vol. 50, no. 12, pp. 2769-2780, Dec. 2002.

[89] Y. Cao, J. J. Xu, V. K. Devabhaktuni, R. T. Ding, and Q. J. Zhang, "An adjoint dynamic neural network technique for exact sensitivities in nonlinear transient modeling and high-speed interconnect design," in *IEEE MTT-S Int. Microw. Symp. Dig.*, PA, Philadelphia, Jun. 2003, pp. 165-168.

[90] T. Liu, S. Boumaiza, and F. M. Ghannouchi, "Dynamic behavioral modeling of 3G power amplifier using real-valued time-delay neural networks," *IEEE Trans. Microw. Theory Techn.*, vol. 52, no. 3, pp. 1025-1033, Mar. 2004.

[91] M. Isaksson and D. W. Ronnow, "Wide-band dynamic modeling of power amplifiers using radial-basis function neural networks," *IEEE Trans. Microw. Theory Techn.*, vol. 53, no. 11, pp. 3422-3428, Nov. 2005.

[92] B. O'Brien, J. Dooley, and T. J. Brazil, "RF power amplifier behavioral modeling using a globally recurrent neural network," in *IEEE MTT-S Int. Microw. Symp. Dig.*, San Francisco, CA, Jun. 2006, pp. 1089-1092.

[93] D. Schreurs, J. Wood, N. Tufillaro, L. Barford, and D. E. Root, "Construction of behavioral models for microwave devices from time domain large signal

measurements to speed up high-level design simulations," *Int. J. RF Microw. Comput.-Aided Eng.*, vol. 13, no. 1, pp. 54-61, Jan. 2003.

[94] H. Sharma and Q. J. Zhang, "Automated time domain modeling of linear and nonlinear microwave circuits using recurrent neural networks," *Int. J. RF Microw. Comput.-Aided Eng.*, vol. 18, no. 3, pp. 195-208, May 2008.

[95] I. A. Maio, I. S. Stievano, and F. G. Canavero, "NARX approach to black-box modeling of circuit elements," in *Proc. IEEE Int. Symp. Circuits Syst.*, Monterey, CA, Jun. 1998, pp. 411-414.

[96] V. Rizzoli, A. Neri, D. Masotti, and A. Lipparini, "A new family of neural network-based bidirectional and dispersive behavioral models for nonlinear RF/microwave subsystems," *Int. J. RF Microw. Comput.- Aided Eng.*, vol. 12, no. 1, pp. 51-70, Jan. 2002.

[97] I. S. Stievano, I. A. Maio, and F. G. Canavero, "Parametric macromodels of digital I/O ports," *IEEE Trans. Adv. Packag.*, vol. 25, no. 5, pp. 255-264, May 2002.

[98] Y. Cao, R. T. Ding, and Q. J. Zhang, "State-space dynamic neural network technique for high-speed IC applications: Modeling and stability analysis," *IEEE Trans. Microw. Theory Techn.*, vol. 54, no. 6, pp. 2398-2409, Jun. 2006.

[99] Sayed Alireza Sadrossadat, Pavan Gunupudi, and Qi-Jun Zhang, "Nonlinear Electronic/Photonic Component Modeling Using Adjoint State-Space Dynamic

Neural Network Technique" accepted in *IEEE Transactions on Components, Packaging and Manufacturing Technology.*

[100] Y. Cao, R. T. Ding, and Q. J. Zhang, "A new nonlinear transient modeling technique for high-speed integrated circuit applications based on state-space dynamic neural network," *IEEE MTT-S Int. Microw. Symp. Dig.*, Fort Worth, TX, Jun. 2004, pp. 1553-1556.

[101] J. M. Zamarreño, P. Vega, L. D. García, and M. Francisco, "State-space neural network for modeling, prediction and control," *Contr. Eng. Practice*, vol. 8, no. 9, pp. 10631075, Sep. 2000.

[102] P. Gil, A. Dourado, and J. O. Henriques, "State space neural networks and the unscented Kalman filter in online nonlinear system identification," *IASTED Int. Conf. Intell. Syst. Contr.*, Tampa, FL, Nov. 2001, pp. 337342.

[103] S. A. Sadrossadat, Y. Cao , and Q. J. Zhang, "Parametric modeling of microwave passive components using sensitivity-analysis-based adjoint neural-network technique," *IEEE Trans. Microw. Theory Techn.*, vol. 61, no. 5, pp. 1733-1747, May 2013.

[104] S. Lum, M. Nakhla, and Q. J. Zhang, "Sensitivity analysis of lossy coupled transmission lines with nonlinear terminations," *IEEE Trans. Microw. Theory Techn.*, vol. 42, no. 4, pp. 607-615, Apr. 1994.

[105] R. Achar and M. S. Nakhla, "Simulation of high-speed interconnects," *Proc. IEEE*, vol. 89, no. 5, pp. 693-728, May 2001.

[106] B. Mutnury,M. Swaminathan, and J. Libous, "Macro-modeling of nonlinear I/O drivers using spline functions and finite time-difference approximation," in *Proc. Electr. Perf. Electron. Packag.*, Princeton, NJ, Oct. 2003, pp. 273-276.

[107] Electronic Design Automation, I/O Buffer Information Specification (IBIS) Ver. 6.0 Sep. 2013 [Online]. Available: http://www.eda.org/ibis/ver6.0/

[108] A. K. Varma, M. Steer, and P. D. Franzon, Improving behavioral IO buffer modeling based IBIS, in *IEEE Trans. Adv. Packag. ,* vol. 31, no. 4, pp. 711721, Nov. 2008.

[109] B. Mutnury, M. Swaminathan, M. Cases, N. Pham, D. N. de Araujo and E. Matoglu, "Macro-Modeling of Nonlinear Transistor-Level Receiver Circuits," in *IEEE Trans. Adv. Packag. ,* vol. 29, no. 1, pp. 55-66, Feb. 2006.

[110] A. Varma, A. Glaser, S. Lipa, M. Steer, and P. Franzon, Simultaneous switching noise in IBIS models, in *Proc. Int. Symp. Electromagnetic Compatibility,* vol. 3, pp. 1000-1004, Aug. 2004.

[111] P. M. Watson and K. C. Gupta, "EM-ANN models for microstrip vias and interconnects in dataset circuits," *IEEE Trans. Microw. Theory Techn.*, vol. 44, no. 12, pp. 2495-2503, Dec. 1996.

[112] J. W. Bandler, M. A. Ismail, J. E. Rayas-Sánchez, and Q. J. Zhang, "Neuromodeling of microwave circuits exploiting space mapping technology," *IEEE Trans. Microw. Theory Techn.*, vol. 47, no. 12, pp. 2417-2427, Dec. 1999.

[113] P. Gunupudi, T. Smy, J. Klein, and Z. J. Jakubczyk, "Self-consistent simulation of opto-electronic circuits using a modified nodal analysis formulation," *IEEE Trans. Adv. Packag.*, vol. 33, no. 4, pp. 979-993, Nov. 2010.

[114] J. Chan, G. Hendry, K. Bergman, and L. P. Carloni, "Physical-layer modeling and system-level design of chip-scale photonic interconnection networks," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems,* vol. 30, no. 10, pp. 1507-1520, Oct. 2011.

[115] Y. Ye, J. Xu, B. Huang, X. Wu, W. Zhang, X. Wang, M. Nikdast, Z. Wang, W. Liu, and Z. Wang, "3-D mesh-based optical network-on-chip for multiprocessor system-on-chip," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 32, no. 4, pp. 584-596, Apr. 2013.

[116] A. Shacham, K. Bergman, and L. P. Carloni, "Photonic networks-on-chip for future generations of chip multiprocessors," *IEEE Trans. on Computers*, vol. 57, no. 9, pp. 1246-1260, Sep. 2008.

[117] G. Hendryy, S. Kamilz, A. Bibermany, J. Chan, B. G. Lee, M. Mohiyuddin, A. Jain, K. Bergman, L. P. Carloni, J. Kubiatowicz, L. Oliker, J. Shalf, "Analysis of photonic networks for a chip multiprocessor using scientific applications," *IEEE International Symposium on Networks-on-Chip*, San Diego, CA, May 2009, pp. 104-113.

[118] A. Joshi, C. Batten, Y.-J. Kwon, S. Beamer, I. Shamim, K. Asanovic, and V. Stojanovic, "Silicon-photonic CLOS networks for global on-chip communi-

cation," *IEEE Int. Symp. Network-on-Chip (NOCS),* San Diego, CA, 2009, pp. 124-133.

[119] J. Chan, G. Hendry, A. Biberman, and K. Bergman, "Architectural exploration of chip-scale photonic interconnection network designs using physical-layer analysis," *J. Lightw. Technol.*, vol. 28, no. 9, pp. 1305-1315, May 2010.

[120] M. Neifeld and W. Chou, "Spice-based optoelectronic system simulation," *Appl. Opt.*, vol. 37, no. 26, pp. 6093-6104, 1998.

[121] B. Whitlock, J. Morikuni, E. Conforti, and S.-M. Kang, "Simulation and modeling: Simulating optical interconnects," *IEEE Circuits Devices Mag.*, vol. 11, no. 3, pp. 12-18, May 1995.

[122] S. Ozyazici and N. Dogru, "Ultrashort pulse generation by spice simulation of gain switching in quantum well laser," *Conf. Lasers Electro-Optics-Pacific Rim, 2007 (CLEO/Pacific Rim 2007)*, Aug. 2007, pp. 1-2.

[123] G. P. Agrawal, *Nonlinear Fiber Optics*, San Diego, CA: Academic Press, 2012.

[124] J. Vlach and K. Singhal, *Computer Methods for Circuit Analysis and Design.* New York: Van Nostrand Reinhold, 1993.

[125] MATLAB. ver. 7.10, The Mathworks Inc., Natick, MA, 2010.

[126] MINIMOS-NT v.2.1. Inst. for Microelectronics, Technical Univ. Vienna, Austria.

[127] J. Psota, J. Miller, G. Kurian, H. Hoffman, N. Beckmann, J. Eastep, and A. Agarwal, "ATAC: Improving performance and programmability with on-chip optical networks." *In Proceedings of IEEE International Symposium on Circuits and Systems,* ISCAS, pages 33253328, 2010.

[128] J. Psota, J. Eastep, J. Miller, T. Konstantakopoulos, M. Watts, M. Beals, J. Michel, K. Kimerling, and A. Agarwal, "ATAC: On-chip optical networks for multicore processors," *Boston Area Architecture Workshop,* Jan. 2007.

[129] T. Smy, P. Gunupudi, S. Mcgarry, and W. N. Ye, "Circuit-level transient simulation of configurable ring-resonators using physical models, *J. Opt. Soc. Am. B,* vol. 28, no. 6, pp. 15341543, Jun 2011.

[130] NXP Semiconductors, [Online]. Available: http://www.nxp.com/