Automatic Neural Network Based Modeling and Its Applications to EM Modeling of Embedded Passives

by

Larry Ton, B. Eng.,

A thesis submitted to the Faculty of Graduate Studies and Research
in partial fulfillment of the requirement for the
degree of Master of Applied Science

Ottawa-Carleton Institute for Electrical and Computer Engineering
Department of Electronics
Carleton University
Ottawa, Ontario K1S 5B6
Canada

©Copyright August 2004, Larry Ton

Library and
Archives Canada

Bibliothèque et
Archives Canada

Published Heritage
Branch

Direction du
Patrimoine de l'édition

395 Wellington Street
Ottawa ON K1A 0N4
Canada

395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence

ISBN: 0-612-97443-X

Our file Notre référence

ISBN: 0-612-97443-X

The author has granted a non-exclusive license allowing the Library and Archives Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque et Archives Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

L'auteur conserve la propriété du droit d'auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

In compliance with the Canadian Privacy Act some supporting forms may have been removed from this thesis.

Conformément à la loi canadienne sur la protection de la vie privée, quelques formulaires secondaires ont été enlevés de cette thèse.

While these forms may be included in the document page count, their removal does not represent any loss of content from the thesis.

Bien que ces formulaires aient inclus dans la pagination, il n'y aura aucun contenu manquant.

Canada

Acknowledgements

I would like to express my sincere thanks to my thesis supervisor Professor Qi-Jun Zhang and co-supervisor Professor M.C.E. Yagoub for their professional guidance, motivation, suggestion, and patience throughout the research work and the preparation of this thesis. Their leadership and vision for quality research and developmental activities has made the pursuit of this thesis a challenging, enjoyable, and stimulating experience. I am highly indebted to them for having trained me into a full-time researcher with technical, computation and presentation skills, and professionalism.

Xiaolei Ding provided me with a lot of help and co-operation during the initial stages of the thesis. Also special thank is given to Jianjun Xu, without his continued discussion, encouragement, and good humor, working on this thesis would have been very tough. I wish to thank my colleagues Hitaish Sharma and Nabil Yazdani for reading the manuscript and for many helpful suggestions to improve the thesis. Lei Zhang, Yi Cao, Vijay K. Devabhaktuni and other students in my group are thanked for the help, friendship, and enjoyable collective working

I am also grateful to Betty Zahalan, Peggy Piccolo, Lorena Duncan, Nagui Mikhail, Scott Bruce and Jacques Lemieux and all other staff and faculty for providing the excellent lab facilities. Financial support from the Department of Electronics through a Teaching Assistantship, MICRONET, NSERC is gratefully acknowledged.

This thesis would not have been possible without years of support and encouragement from my parents, especially my mother, who brought me up and always proud of me so this thesis is dedicated to her. Last but not least, I would like to thank my family, friends and my girlfriend for their love, support and encouragement have been and will be the source of strength for overcoming any difficulty and achieving success throughout my whole life.

Abstract

The electronics industry for manufacturability-driven design and time-to-market demands powerful and efficient computer-aided design (CAD) techniques. In recent years, a CAD approach based on neural networks has been introduced for microwave modeling, simulation, and optimization. Though proven successful in many cases where other traditional techniques failed to give satisfactory results, there are still many issues needed to be solved. We need to address difficulties with the correct choice of network parameters, like the amount of sampling data to train the neural network model, initial set of weights, adequate network size, and structure. Key objectives of this thesis are: (a) To formulate efficient neural network modeling algorithms that can facilitate automatic generation of accurate RF and microwave neural models, and (b) To develop robust neural modeling techniques that would enable everyday users to learn, apply and gain experience with neural network without a special expertise in this field.

Major contributions of the thesis include the Automatic Multilayer Selection (AMS) technique and the combination of the proposed AMS technique with an existing automatic model generation (AMG) algorithm to automatically develop a compact neural network model. A microwave component model can be created starting with a simple neural network structure and then proceeding with neural network training in a systematic manner. During each stage, the AMS algorithm utilizes the training error criteria to automatically adjust the number of layers or the number of neurons in each layer of the neural network structure and consequently uses AMG algorithm to train a model to meet

a user-desired accuracy. By combining the proposed AMS with Automatic Data Generation (ADG) and AMG algorithms, a more efficient and automated modeling framework is developed to generate neural network models that accurately match training data, with minimal human intervention.

Table of Contents

Abstract.....	i
Table of Contents.....	iii
List of Figures.....	vii
List of Tables	x

Chapter 1

Introduction.....	1
1.1 Background and Motivation	1
1.2 Thesis Objectives and Contributions	5
1.3 Organization of the Thesis	6

Chapter 2

Artificial Neural Network for RF and Microwave	8
2.1 Introduction to the Neural Based Modeling.....	8
2.2 Overview of Neural Network Based Modeling Techniques Applied to Embedded Passive Component Model	10
2.2.1 Collection Training Data.....	12
2.2.2 Neural Network Structures Selection.....	14
2.2.2.1 Multi-Layer Perceptrons (MLP).....	14
2.2.3 Neural Network Training Algorithms.....	18
2.2.3.1 Training Objective.....	18

2.2.3.2 Backpropagation Algorithm	19
2.2.3.3 Conjugate Gradient Training Algorithm	20
2.2.3.4 Quasi-Newton Training Algorithm	22
2.2.4 Testing the Neural Network Model	23
2.3 Conventional Neural Network Model Development.....	24
2.3 Conclusions.....	27

Chapter 3

Automatic Neural Network Based Modeling Techniques Applied to Embedded Passive Component Model	28
3.1 Introduction	28
3.2 Automatic Neural Network Based Modeling Algorithms.....	30
3.2.1 Automatic Data Generation	30
3.2.2 The Key Aspects of Automatic Data Generation	31
3.2.3 Verification of Automatic Data Generation Algorithm	33
3.3 Proposed Automatic Multilayer Selection.....	36
3.3.1 Introduction.....	36
3.3.2 The AMS Algorithm	38
3.3.3 Discussion.....	41
3.3.3.1 Minimize the Possible Training Structures	40
3.3.3.1.1 Choice of the Number of Layers.....	44
3.3.3.1.2 Minimize the Neural Network Topologies	46

3.3.3.2 Speeding Up the Neural Network Model Training Using Newton-Raphson Method	51
3.3.3.3 Optimize the Neural Network Structure.....	57
3.3.3.4 Combination of Newton-Raphson Method and Binary Search Algorithms	58
3.3.3.5 Converting the Prediction of Total Weight to an Appropriate Network Structure	61
3.4 Overall Automation of Neural Network Model Generation	63
3.4.1 Combination AMG, ADG and AMS Algorithms	63
3.4.2 Details Expanding of Overall Automatic Neural Network Model Generation.....	65
3.5 Conclusions	69

Chapter 4

Applying Automatic Model Generation for Developing Example of Embedded Passive Component Models.....	70
4.1 Introduction	70
4.2 Development of Embedded Resistor Neural Network Model Using Advanced AMG Approach versus Conventional Method	71
4.3 Development of 2-Loop Spiral Inductor Neural Network Model Using Advanced AMG Approach	75
4.4 Development of the Combined Equivalent Circuit and Neural Network Model for Embedded Capacitor Using the Advanced AMG Approach	81

4.5 Applying Embedded Components in Frequency Domain Circuit Simulation..	83
4.6 Applying Embedded Components in Time Domain Circuit Simulation	87

Chapter 5

Conclusions and Future Research.....	90
5.1 Conclusions	90
5.2 Future Directions.....	91
Bibliography	95

List of Figures

Figure 2.1	The user defined neural network model structure for an embedded capacitor. (a) 3D representaion of an embedded capacitor model and (b) The neural network model structure for an embedded capacitor.....	11
Figure 2.2	Illustration of the feedforward multilayer perceptrons (MLP) structure. Typically, the neural network consists of one input layer, one or more hidden layers, and one output layer.	15
Figure 2.3	Flow charts of the conventional steps involved in neural network model development.....	26
Figure 3.1	Systematic framework of the proposed automatic data generation algorithm in the forms of a flow-chart.	32
Figure 3.2	The total number of weights is sorted in order from smallest to largest. A model with smallest size will be chosen for neural network training. If the desired accuracy is not achieved the AMS automatically selects the next larger structure and continue training of that model.....	40
Figure 3.3	3D and 2D layout of the embedded components using Sonnet-Lite. (a) An embedded resistor, (b) An embedded capacitor and (c) An embedded inductor.	43
Figure 3.4	Comparison percentage training error versus different neural network topologies of MLP3, MLP4, and MLP5. (a) An embedded resistor, (b) An embedded capacitor, (c) An embedded inductor.	46

Figure 3.5	Comparison the total weight versus the training error after eliminate non-efficient structures. (a) An embedded resistor, (b) An embedded capacitor, and (c) An embedded inductor.....	53
Figure 3.6	Applying Newton-Raphson extrapolation method to calculate the approximate weights for the next neural network model training	54
Figure 3.7	Demonstration of using Newton-Raphson algorithm.....	59
Figure 3.8	Demonstration of using Binary search Algorithm.....	60
Figure 3.9	Systematic framework of the proposed combination of AMG and AMS algorithms in the form of a flow chart	68
Figure 4.1	The outputs of the neural network model for the S-parameters of embedded resistors match very well with EM test data which are never used in training process.....	74
Figure 4.2	Comparison of neural network model prediction of S-parameters with original EM <i>Sonnet-Lite</i> data for the embedded inductor. Three out of 27 different inductor lengths and 100 frequency samples per length are shown above ..	80
Figure 4.3	(a) 3D physical layout of an embedded capacitor and (b) The neural network model combined with equivalent circuit.....	82
Figure 4.4	Comparison of RS_{II} of the original EM data (-x-) and the equivalent circuit neural network model.....	83
Figure 4.5	The amplifier circuit, in which, all the passive components are represented by the pure neural network models. N_R and N_C denote neural models of embedded resistors and capacitors respectively.....	84

Figure 4.6	Gain vs. Frequency for <i>ADS</i> lumped components (—○—) and the circuit response after optimization (—).....	86
Figure 4.7	Illustration of VLSI interconnect example for signal integrity analysis in printed circuit board with coupled transmission line, nonlinear buffers and embedded passives.....	87
Figure 4.8	Comparison of signals from input buffer and output buffer of the signal integrity example before and after optimization of the embedded passives.....	89

List of Tables

Table 3.1	Different neural network topologies having same total weight.....	45
Table 3.2	Comparison training error of different neural network topologies for MLP4 (a) An embedded resistor, (b) An embedded capacitor, (c) An embedded inductor	49
Table 3.3	Training errors sorted after eliminating non-efficient structures for three embedded passives models	52
Table 4.1	Neural network model training of the embedded resistors using conventional approach.....	73
Table 4.2	Neural network model training of the embedded resistors using new AMG approach.....	74
Table 4.3	Comparison of the time taken by the AMG method and manual step-by-step neural modeling approach for the embedded resistors.	75
Table 4.4	The training stage for the spiral inductors using combination of AMG and AMS method.....	79
Table 4.5	The comparison of geometry of embedded passives before and after optimization in the amplifier circuit.	86
Table 4.6	Comparison of the physical/geometrical parameters in embedded passives before optimization and after optimization.....	88

Chapter 1

Introduction

1.1 Background and Motivation

Product trends in the wireless communications market include higher frequencies, smaller physical size, faster time-to-market cycles, and lower cost. RF modules with embedded passive components offer a possible solution to meet these new requirements. Higher frequency operation, smaller physical size, and faster design cycles can result from placing and interconnecting RF passive components inside the RF modules rather than placing and interconnecting individual surface mount components on printed circuit boards [1].

RF modules with embedded passives are designed using metal structures on internal substrate layers to build and interconnect RF passive components [2]. Because of the lack of detailed embedded component specifications, models often result in a trial and error development process. A cycle of designing, building and testing an RF module can be repeated several times before the product meets specifications. This is a very tedious and time consuming process and is heavily dependant on the designer's skills, speed and experience. With the effective use of computer aided design (CAD) tools in both

electrical and physical design stages, one can shrink design margins and reduce the number of design iterations, however this is typically very computationally intensive. It has been well recognized that the CPU time and memory space required to simulate a fully integrated microwave circuit, using electromagnetic (EM) simulators far exceeds the capabilities of today's computer workstations.

The efficiency of RF/microwave CAD tools relies heavily upon the speed and accuracy of component models [3][4]. The design of embedded components (resistors, inductors and capacitors) is non-trivial due to electromagnetic interactions that lead to parasitic effects, leading to non-ideal frequency behavior. The conventional approach for embedded model design requires an equivalent circuit to capture the responses of embedded passives. However, the equivalent circuit method may not be accurate enough to reflect high frequency EM effects. Detailed physics-based models are very accurate [5] but are very slow. Table look-up models can be relatively fast, but suffer from the disadvantages of large memory requirements and limitations in parameter dimensionality [6]. For high-level optimization of circuit parameters, the frequency and geometrical/electrical model parameters are continuously swept through the parameter space. Therefore, efficient modeling techniques that can provide such continuous variations are essential.

In past years, there has been a strong interest among researchers in using Artificial Neural Network (ANN) in microwave model development and design. As such, ANN has been recognized as a useful tool for RF and microwave modeling and design [6][7]. ANN

models can be trained from physics/EM simulation or measurement data and subsequently used during circuit analysis, optimization, and design. The trained models are fast and can represent physics/EM behaviors previously learned which would be otherwise computationally expensive. The learning ability of ANN is very useful when analytical models for a new device are not available, i.e., modeling of an embedded capacitor or resistor. ANN networks can be more accurate than polynomial regression models and empirical models [8], handle more dimensions than look-up tables [9], easier to develop when a new device/technology is introduced [10], allow more automation in model development than conventional circuit models [11], and can also generalize meaning the model can respond to new data that has not been used during training [6]. ANN has been successfully used in a variety of applications such as modeling and optimization of high-speed circuits.

Reliable CAD solutions need accurate neural network models. However developing an accurate neural network model may require a lot of human effort and experience. In high-level circuit design, the component models should be continuously varied both with frequency, geometrical and/or electrical parameters [11]. In order to develop continuous neural network embedded component models, one has to first create a model structure for each of the geometries, then use EM simulators to generate the training data. While data generation can be done by computers, model structure creation and neural model training is often done by humans. The conventional process is to use human experience and skills to create an embedded model structure and to train a neural network model to match EM data. This process is trial and error based and is very human intensive. With the rapid

development in microwave technology, new devices constantly evolve, and new models are always required. Manually modifying the existing models to match new data is very inefficient. Methods that can automatically generate a new model to adapt with new data are much more desirable. To find a systematic and automatic embedded model generation process is one of the motivations of this thesis.

As the signal frequency increases, the dimensions of embedded passives in multi-layer circuits become a significant fraction of signal wavelength. The conventional time/frequency domain electrical models for the components are no longer accurate. Models with continuous physical/geometrical information must include EM effects [11]. Recently the ANN approach has achieved great success in microwave/RF circuit applications and statistical design. This is mainly attributed to the ability of ANN's to model the passive components used in the circuits, and achieve nearly the same degree of accuracy as can be afforded by EM simulations. ANN's have also demonstrated the capability to generalize and to predict accurate model parameters for data outside the training set. These properties make ANN models suitable for circuit applications and statistical design. However, such ANN models trained to learn S-parameter data cannot be used directly in time domain circuit simulation and optimization. How to utilize the existing techniques for automatic development of reliable embedded models that can be efficiently used in both of time and frequency-domain CAD tools is another motivation in this thesis.

1.2 Thesis Objectives and Contributions

The main objective of this thesis is to develop an advanced ANN modeling technique which can automatically generate an efficient and compact neural network structure for a highly non-linear model such as an embedded passive component. The proposed Automatic Multilayer Selection (AMS) algorithm uses the training error to dynamically adjust the number of layers or the number of neurons in each layer of the neural network structure to produce a compact network size for a given microwave problem. A unified framework (advanced AMG) incorporating the proposed technique and the existing AMG [12] algorithm is developed. This can facilitate automatic generation of microwave neural network models of highest accuracies with a compact network size. To validate the new proposed methods, neural models of embedded passive components were generated. The proposed methods outperformed the conventional model methods in terms of accuracy, neural network size and generation time. This validates the proposed methods as a better and more feasible alternative over the conventional methods.

The other objective of this thesis is to apply the EM based neural network model created by the proposed advanced AMG in frequency/time domain circuit simulators. Neural models trained from EM data of microwave components act as a bridge connecting EM simulation with computer-aided design (CAD) tools. Through fast, accurate and continuous EM-based neural models of embedded components, we enable consideration of EM effects in high-frequency and high-speed CAD, including components geometrical/physical parameters as optimization variables. Examples of amplifier circuit

simulation and board-level interconnect examples [13][14] are developed to validate our models.

The contributions of the thesis include the Automatic Multilayer Selection (AMS) technique and the combination of the proposed AMS technique with an existing automatic model generation (AMG) algorithm to automatically develop a compact neural network model. A microwave component model can be created starting with a simple neural network structure and then proceeding with neural network training in a systematic manner. During each stage, the AMS algorithm utilizes the training error criteria to automatically adjust the number of layers or the number of neurons in each layer of the neural network structure and consequently uses AMG algorithm to train a model to meet a user-desired accuracy. By combining the proposed AMS with Automatic Data Generation (ADG) and AMG algorithms, a more efficient and automated modeling framework is developed to generate neural network models that accurately match training data, with minimal human intervention.

1.3 Organization of the Thesis

The thesis is organized as follows.

In Chapter 2, an overview of ANN-based RF and microwave modeling including a literature review is presented. Commonly used neural network structures, training algorithms, data sampling and scaling strategies, and various key issues in automation of ANN-modeling are described.

In Chapter 3 the new concept of Automatic Multilayer Selection (AMS) is introduced. A unified framework combining the proposed AMS with an existing Automatic Data Generation (ADG), and Automatic Model Generation is developed to create more advanced Automatic Model Generation techniques. The new efficient AMG techniques are aimed at generation of compact microwave neural models of highest possible accuracy without intensively using human labor and understanding of neural network modeling issues.

In Chapter 4, an object-oriented implementation of the new AMG algorithm is accomplished in C++. The computer program has been used in deriving the results in this thesis and incorporated into a trial version of *Neuromodeler Plus* [15] software. With the help of the software, we are able to apply the advanced AMG technique in developing the embedded passive component models. The proposed technique is validated through the demonstration of creating a set of embedded passives (resistors, capacitors, and inductor) models to be used in design of multi-layer Printed Circuit Boards. Advantages of the new AMG algorithm relative to conventional neural modeling approaches are illustrated through these examples.

Finally, conclusions of the thesis are presented in Chapter 5 and recommendations for applying AMG in future research towards accomplishing end-to-end neural based RF and microwave CAD solutions are also made.

Chapter 2

Artificial Neural Network for RF and Microwave

2.1 Introduction to the Neural Based Modeling

The rapid development in telecommunication, VLSI and wireless industries over the last decade have lead to many circuit design and analysis problem with a large and computationally expensive. This has resulted in an explosion of interest for improving the Computer Aided Design (CAD) tools [11]. Today most of CAD tools have difficulties simulating the Electromagnetic (EM) and Physics of passive components. The reason is that the time-consuming nature of theoretical EM/physics of high-level microwave-CAD operations (i.e., Monte Carlo analysis and yield optimization) prohibitively expensive in terms of CPU-time. The efficiency of RF/microwave CAD tools relies heavily upon the speed and accuracy of component models [6]. But modeling is still remains a major bottleneck for the successful CAD of high-frequency components and circuits. A variety of modeling approaches are available for passive components. Some of them are fast but limited accuracies or exhibit limited degree of nonlinearity and hence not suited for RF/Microwave modeling. Some of them are accurate but slow as they involve numerical simulations based on extensive theory. Therefore development of

accurate and fast RF and microwave models is essential for robust high-frequency CAD design.

Recently, neural network technology has been introduced in the microwave modeling, simulation, and optimization. Neural network are information processing systems inspired by the ability of human-brain to learn from observations and to generalize by abstraction [6]. Neural network possess the distinguished ability of learning from samples of input-output data to accurately model nonlinear relationships. Fast and accurate neural network models can be developed from measured or simulated RF/microwave data through a learning process called training. Resulting neural network models once developed can be used in place of computationally intensive physical/EM model of active and passive components [4][16]. Recent work by microwave researchers demonstrated the ability of neural network to accurately model a variety of microwave components, such as microstrip interconnects [16], spiral inductors [17], embedded components [18][19][20], vias [21], FET devices [16][22], HBT devices [23], HEMT devices [24], filter [25], amplifier [4][26], coplanar waveguide (CPW) circuit component [27], mixer [26], antennas [28], and, packaging and interconnects [29], etc. neural network have also been used in circuit simulation and optimization [4][19][30], signal integrity analysis and optimization of VLSI interconnects [19][29][31], microstrip circuit design [32], process design [33], synthesis [34], and microwave impedance matching [35]. These pioneering works have established the framework of neural modeling technique for both device and circuit level in microwave application.

2.2 Overview of Neural Network Based Modeling Techniques Applied to Embedded Passive Component Model

Let x represent an n -vector containing the input physical parameters of a microwave device, such as the length, thickness, material of an embedded capacitor and signal frequency. Let y represent an m -vector containing the outputs responses of the device, such as S-parameters of an embedded capacitor. The EM/physics relationship between y and x can be represented as

$$y = y(x) \quad (3.1)$$

The EM/physics relationship between x and y is highly non-linear and multi-dimensional. The theoretical model for $y(x)$ may be too complicated to implement and use or the theoretical model may be computationally too intensive for online microwave design and repetitive optimization (e.g., 3D full-wave EM analysis inside a Monte Carlo statistical design loop). The objective is to develop a model that will represent the $x - y$ relationship more efficiently. Figure 2.1 illustrates the embedded model of a capacitor and the neural network model structure to represent it.

A fast neural model is developed by training a neural network with a set of measured/simulated samples $\{(x_q, y_q), q \in L\}$, where y_q represents measured/simulated output for sample input x_q , and L represents the set of training data. Let the input-output mapping by the neural network be defined as

$$\tilde{y} = \tilde{y}(x, w) \quad (3.2)$$

Figure 2.1 The user defined neural network model structure for an embedded capacitor:
 (a) 3D representation of an embedded model of a capacitor and (b) The neural network model structure for an embedded capacitor. Where L is a length, T is a thickness, ϵ_{rc} is a dielectric constant of an embedded capacitor and f is signal frequency.

Where $\tilde{y}$ is an m -vector of neural model outputs and w is a weight vector containing the adjustable parameters inside the neural network. The objective of training is to adjust w such that the error between neural model output $\tilde{y}$ and training data y is minimized. Once trained, the neural network model can be used for predicting the output values given only the values of the input variables. In the model testing stage, an independent set of input-output samples, called testing data issued to test the accuracy of the neural model. The ability of neural models to predict y when presented with input parameter values x , never seen during training is called the generalization ability. A trained and tested neural model can then be used online during microwave design stage providing fast model evaluation replacing original slow EM/Device

simulators. The benefit of the neural model approach is especially significant when the model is highly repetitively used in design process such as, optimization, Monte Carlo analysis and yield maximization. In the next section a detailed review of neural network model development is presented.

2.2.1 Collection Training Data

The first important step towards neural network embedded component model development is to collect the EM data of an embedded component. Theoretically, neural network models can be consider as a black box models, whose accuracy depend on the data presented to it during training. Before collecting data to build a black box we need to know the inputs and outputs of that black box. For the embedded passive components the inputs to the black box are the geometrical and physical parameters (e.g., Length, Width) and the outputs from black box are the real and imaginary parts of S-parameters. Once all the inputs and outputs have been determined the next step is to define the range of training, test and the distribution of the data samples within that range. The range of the input space in which the neural network model would be used after training is $[x_{\min}, x_{\max}]$. Training data is sampled slightly beyond the neural model utilization range, i.e., $[x_{\min} - \Delta, x_{\max} + \Delta]$, in order to ensure reliability of the neural model at the boundaries of that range. Test data is generated in the range $[x_{\min}, x_{\max}]$. When the range of model input parameters is finalized a sampling distribution needs to be chosen. Commonly used sample distributions include uniform and non-uniform grid distributions [6]. In uniform grid distribution, each input parameter x_i is sampled at equal intervals. In non-uniform grid distribution each input parameter is sampled at unequal intervals. For the problem

with a highly nonlinear in certain sub-regions of the x -space, it requires dense sampling in such sub-regions. In this case non-uniform grid distribution usually works better than a uniform.

A complete process of collecting data for neural network training requires three sets of data: training data, validation data and testing data. Training data is used to train the neural models updating the neural network training weights. Validation data is required to monitor the training quality of the neural network model. This will give an indication to terminate the training. Test data is used to determine the final quality of the models generated.

The training, validation and testing data can come either from detailed software simulator or from a measurement setup. The most common use simulators such as *MINIMOS* [36], *Sonnet-Lite* [37], *Ansoft-HFSS* [38], *HSIPCE* [39], *Agilent-ADS* [40], *Ansoft-Ensemble* [41], and measurement equipment as vector or scalar network analyzers. Usually the measurement data from actual components can be very costly compare to obtain data from software simulator. When the data is hard to obtain, the generated data can be used for training the neural network and for testing the resulting neural model. In practice, both simulations and measurements can have small errors. While errors in simulation can be due to truncation/round-off or non-convergence, errors measurement can be due to equipment limitations or tolerances [42]. Hence, it is an essential to generate as many data sets as possible or do a lot of measurements to limit the error in data collection.

2.2.2 Neural Network Structures Selection

The next major step in neural network development is to identify a suitable neural network structure for a model that we wish to design. This is not trivial due to the size of the structure, i.e., the number of neurons, and the number of hidden layers, are not easy to determine or predict in the development of a neural network. The problems of under-learning and over-learning have to be deal with, i.e., small size neural network cannot learn the problem very well (under-learning). The large size structure the neural network model can match the training data very well but does not match with the validation data (over-learning). The reason is that too many hidden neurons with insufficient training data that lead to too much freedom in the input-output relationship represented by a neural network. Hence, it may require many trial of error in order to select a neural network structure that would give a model accuracy match with user desire. There are varieties of neural network structures have been developed in the neural network community for microwave circuit, signal processing, control and so on. In this thesis we are only focus on the Multilayer Perceptrons (MLP) neural network structures that is used in automatic neural network model development.

2.2.2.1 Multi-Layer Perceptrons (MLP)

One of the most widely use feedforward neural network structure is the Multilayer Perceptrons (MLP). Feedforward neural network are a basic type of neural network capable of approximating generic continuous and integrable ones. Typically, the MLP neural network consists of an input layer, one or more hidden layer and an output layer,

as shown in Figure 2.2. Suppose the total number of hidden layers is N_h . The input layer is layer 0, Let the number of neurons in hidden layer l be $N_l, l=1,2,\dots,N_h$. Let w_{ij}^l

Figure 2.2 Illustration of the feedforward multilayer perceptrons (MLP) structure. Typically, the neural network consists of one input layer, one or more hidden layers, and one output layer.

represent the weight of the link between j^{th} neuron of $(l-1)^{th}$ hidden layer and i^{th} of l^{th} hidden layer, the θ_i^l be the bias parameter of i^{th} neuron of l^{th} hidden layer. Let x_i represents the i^{th} input parameter to the MLP.

Let $\tilde{y}_i^l$ be the output of i^{th} neuron of l^{th} hidden layer, which can be computed according to the standard MLP formulae as

$$\tilde{y}_i^l = \sigma\left(\sum_{j=1}^{N_{l-1}} w_{ij}^l \cdot \tilde{y}_j^{l-1} + \theta_i^l\right), \quad l = 1, 2, \dots, N_L, \quad i = 1, 2, \dots, N_h \quad (2.1)$$

$$\tilde{y}_i^0 = x_i, \quad i=1, 2, \dots, N_x, \quad N_x = N_0 \quad (2.2)$$

where $\sigma(\cdot)$ is the activation function of hidden neurons. Let v_{ki} represent the weight of the link between i^{th} neuron of N_h^{th} hidden layer and k^{th} neuron of output layer, and β_k be the bias parameter of k^{th} output neuron. The outputs of MLP can be computed as

$$\tilde{y}_k = \sum_{i=1}^{N_{N_h}} v_{ki} \cdot \tilde{y}_i^{N_h} + \beta_k, \quad k=1, 2, \dots, N_y. \quad (2.3)$$

For function approximation, output neurons can be processed by linear function as shown in (2.3). The most commonly used activation function $\sigma(\cdot)$ for hidden neurons is the logistic sigmoid function given by

$$\sigma(\gamma) = \frac{1}{(1 + e^{-\gamma})} \quad (2.4)$$

which has property

$$\sigma(\gamma) \rightarrow \begin{cases} 1 & \text{as } \gamma \rightarrow +\infty \\ 0 & \text{as } \gamma \rightarrow -\infty \end{cases} \quad (2.5)$$

Other possible candidates for $\sigma(\cdot)$ are the arctangent function

$$\sigma(\gamma) = \left(\frac{2}{\pi} \right) \arctan(\gamma) \quad (2.6)$$

and the hyperbolic tangent function

$$\sigma(\gamma) = \frac{(e^\gamma - e^{-\gamma})}{(e^\gamma + e^{-\gamma})} \quad (2.7)$$

All these functions are bounded, continuous, monotonic and continuously differentiable.

The universal approximation theorem states that for any arbitrary nonlinear, continuous, multi-dimensional function, there always exists a 3-layer perceptrons that will approximate that to any accuracy desired by the user [43][44]. However, the universal approximation theorem does not tell us the number of neurons needed. Therefore, failure to develop an accurate neural model can be attributed to inadequate learning, inadequate number of hidden neurons, or the presence of stochastic rather than a deterministic relationship between input and output [44]. In practice, the number of hidden neurons depends on the degree of nonlinearities and the dimensions of the original problem. But too many hidden neurons will lead to over-learning in the training process. Neural network with one or two hidden layers, i.e., 3-layer or 4-layer MLP are more frequently used and are usually suitable for RF/microwave application. The performance of neural network can be evaluated in terms of generalization capability and mapping capability. In [45], Tamura demonstrated that a 3-layer MLP is preferred in function approximation

where generalization capability is a major concern. Intuitively, 4-layer MLP would perform better in nonlinear problems in which localized behavioral components exist repeatedly in different regions of the problem space.

2.2.3 Neural Network Training Algorithms

2.2.3.1 Training Objective

A neural network model can be developed through an optimization process called training. Let us consider the training data includes N_p sample pairs, $\{(x_p, d_p), p = 1, 2, \dots, N_p\}$, where x_p and d_p are N_x and N_y dimensional vectors representing the inputs and outputs of the neural network respectively. Let w represent the parameters inside the neural network called as the weight vector. Let $\tilde{y} = f_{ANN}(x, w)$ represent the input-output relationship of the neural network. The objective of training is to find w such that the error between neural network predictions and the desired output are minimized

$$\min_w E(w) = \sum_{i=1}^{N_y} e_i(w) = \frac{1}{2} \sum_{i=1}^{N_y} (\tilde{y}_{pk} - d_{pk})^2 \quad (2.8)$$

where d_{pk} is the k^{th} element of vector d_p , $\tilde{y}_{pk}$ is the k^{th} output of the neural network when the input is x_p . The objective of neural network training is to adjust neural network connection weights w such that $E(w)$ is minimized.

The objective function $E(w)$ is a nonlinear function w.r.t. the adjustable weight vector w . Due to the complexity of $E(w)$, iterative algorithms are often used to explore the parameter space efficiently.

2.2.3.2 Backpropagation Algorithm

Backpropagation [46] is the most popular algorithm for neural network training. In each step of the step-by-step algorithms, the BP is first done layer by layer to calculate the derivation of cost function $E(w)$ to weights w . The weights of the neural network w then are updated along the negative gradient direction in the weight space. The update formula are given by

$$\Delta w_{now} = w_{next} - w_{now} = -\eta \frac{\partial E(w)}{\partial w} \Big|_{w=w_{now}} \quad (2.9)$$

$$\Delta w_{now} = w_{next} - w_{now} = -\eta \frac{\partial e_i(w)}{\partial w} \Big|_{w=w_{now}} \quad (2.10)$$

where constant η called learning rate controls the step size of weight update, w_{next} and w_{now} denotes the next and present value of w respectively. In Equation (2.10), the weights are updated after all the training samples have been used to teach the neural network. Update Equation (2.9) is called batch mode update, where the weights are updated after all training samples have been presented to the network.

The learning rate η is a sensitive parameter in the back propagation algorithm. If η is small number, the more iteration is needed for training. However, if η is too large to speed up the learning process, the training may be unstable due to weight oscillation [47]. The addition of a momentum term to weight update Equation in (2.9) and (2.10) as proposed by [46], provided significant improvement to the BP, reducing the weight oscillation.

$$\Delta w_{now} = -\eta \frac{\partial E(w)}{\partial w} \Big|_{w=w_{now}} + \alpha (w_{now} - w_{old}) \quad (2.11)$$

$$\Delta w_{now} = -\eta \frac{\partial e_i(w)}{\partial w} \Big|_{w=w_{now}} + \alpha (w_{now} - w_{old}) \quad (2.12)$$

where the constant α is the momentum factor which controls the influence of the last weight update direction on the current weight update, w_{old} represents the last point of w . To reduce the weight oscillation, many approaches have been proposed, such as invoking a correction term that uses the difference of gradients [48], and an extra constrain is place so that the alignments of successive weight updates are maximized [49].

An important way to efficient train a neural network by BP is to use adaptation schemes that allow the learning rate and the momentum factor to be adaptive during learning [47], e.g., adaptation according to training errors [50]. The adaptation is determined from two factors, one being the current derivative of the training error with respect to the weights, and the other being an exponentially weighted sum of the current and past derivatives of the training error. In [51], which is considered as an extension of Jacob's heuristics, corrects the values of weights near the bottom of the error surface ravine with a new acceleration algorithm. This correction term uses the difference between gradients to reduce the weight oscillation during training.

2.2.3.3 Conjugate Gradient Training Algorithm

The conjugate gradient methods are originally derived from quadratic minimization and the minimum of the objective function E_T , can be efficiently found within N_w iterations [52] .

With initial gradient $g_{initial} = \frac{\partial E}{\partial w} \Big|_{w=w_{initial}}$, and direction vector $h_{initial} = -g_{initial}$, the conjugate gradient method recursively constructs two vector sequence,

$$\mathbf{g}_{next} = \mathbf{g}_{now} + \lambda_{now} \mathbf{H} \mathbf{h}_{now} \quad (2.16)$$

$$\mathbf{h}_{next} = -\mathbf{g}_{now} + \gamma_{now} \mathbf{h}_{now} \quad (2.17)$$

$$\lambda_{now} = \frac{\mathbf{g}_{now}^T \mathbf{g}_{now}}{\mathbf{h}_{now}^T \mathbf{H} \mathbf{h}_{now}} \quad (2.18)$$

$$\gamma_{now} = \frac{\mathbf{g}_{next}^T \mathbf{g}_{next}}{\mathbf{g}_{now}^T \mathbf{g}_{now}} \quad (2.19)$$

or,

$$\gamma_{now} = \frac{(\mathbf{g}_{next} - \mathbf{g}_{now})^T \mathbf{g}_{next}}{\mathbf{g}_{now}^T \mathbf{g}_{now}} \quad (2.20)$$

where $\mathbf{h}$ is called the conjugate direction and $\mathbf{H}$ is the Hessian matrix of the objective function E . λ and γ are called learning rate, and subscription *now*, *next* are the current and next values of $\mathbf{g}$, $\mathbf{h}$, λ , γ respectively. Here, (2.19) is called the Fletcher-Reeves Equation and (2.20) is called the Polak-Ribiere formula. To avoid the need of hessian matrix to compute the conjugate direction, we proceed from $\mathbf{w}_{now}$ along the direction $\mathbf{h}_{now}$ to the local minimum of E_{Tr} at $\mathbf{w}_{next}$ through line minimization, and then set

$\mathbf{g}_{next} = \frac{\partial E}{\partial \mathbf{w}} \Big|_{\mathbf{w}=\mathbf{w}_{next}}$. This $\mathbf{g}_{next}$ can be used as the vector of (2.16), and as such (2.18) is no

longer needed. We can make use of this line minimization concept to find conjugate direction in neural network training, thus avoiding intensive Hessian matrix computations. In this method, the descent direction is along the conjugate direction which can be accumulated without computations involving matrices. As such, conjugate gradient methods are very efficient and scale well with the neural network size.

2.2.3.4 Quasi-Newton Training Algorithm

Quasi-Newton algorithm [47] is also derived from quadratic objective function optimization. The inverse of Hessian matrix $A = H^{-1}$ is used to bias the gradient direction. In Quasi-Newton training method, the weights are updated by

$$w_{next} = w_{now} - \eta A_{now} g_{now} \quad (2.21)$$

The matrix A here is not computed. It is successively estimated using Broyden-Fletcher-Goldfarb-Shanno (BFGS) Equation [53] as

$$A_{now} = A_{old} + \left(1 + \frac{\Delta g^T A_{old} \Delta g}{h^T \Delta g}\right) \frac{h h^T}{h^T \Delta g} - \frac{h^T \Delta g A_{old} + A_{old} \Delta g h^T}{h^T \Delta g}, \quad (2.22)$$

where

$$h = w_{now} - w_{old}$$

and

$$\Delta g = g_{now} - g_{old} \quad (2.23)$$

Again, we use subscript *old*, *now*, *next* to represent the old, the current and the new of A , g , w respectively.

Quasi-Newton algorithm exhibits faster convergence than conjugate gradient method. However, standard quasi-Newton methods require N_w^2 storage space to maintain an approximation of the inverse Hessian matrix and a line search is indispensable to calculate a reasonably accurate step length, where N_w is the total number of weights in the neural network structure. Limited-memory (LM) or one-step BFGS is a simplification

in which the inverse Hessian approximation is reset to the identity matrix after every iteration, thus avoiding the need to store matrices [54][55] .

2.2.4 Testing the Neural Network Model

This is the final step of the model generation. This step is performed after the models are generated. This step ensures the quality of the model produced. Depending upon the quality, the model needs to be re-trained if the testing error is too high. An independent set of data known as the test data is required for this purpose. A quantity δ_{pk} is defined as

$$\delta_{pk} = \frac{\tilde{y}_k(x_p, w) - d_{pk}}{d_{k,max} - d_{k,min}}, k=1, \dots, N, p \in T_E \quad (2.24)$$

A quality criterion based on the q^{th} norm is then defined as

$$M_q = \left[\sum_{p \in T_E} \sum_{k=1}^{N_y} |\delta_{pk}|^q \right]^{\frac{1}{q}} \quad (2.25)$$

When $q = 1$, the average test error can be calculated directly from M_1 as

$$\text{Average Test Error} = \frac{M_1}{N_{TE} N_y} \quad (2.26)$$

where N_{TE} is the number of samples in test data set and N_y is the number of neural model outputs. When $q = 2$, the q^{th} norm is the Euclidean distance between neural model prediction and test data. When $q = \infty$, the q^{th} norm measure is the maximum test

error, which is also called as the worst-case error among entire test data and all model outputs.

2.3 Conventional Neural Network Model Development

The conventional neural network model development involves various steps such as collection of the training data, selecting neural network structures, neural network training and neural model testing [6]. A key to obtain an accurate model is a good collection of the training data, which is well distributed, sufficient, and accurately measured/simulated.

Training data collection may be very expensive in the reality of microwave problem. To minimize the number of training data requires a careful of selecting neural network structure would help to achieve higher model accuracy with few training data [56]. For example, a feed-forward neural network with smooth switching functions in the hidden layer is good at modeling smooth, slowly, varying nonlinear functions, while a feed-forward neural network with Gaussian functions in the hidden layer could be more effective in modeling nonlinear functions with large variations. Furthermore, for knowledge based neural network, less training data are required as well as accuracy can be achieved [16].

With an accurate training data obtained and an appropriate neural network selected, a neural network model still can not guarantee to give a good model unless it trained by a suitable training algorithm. Training is an essential step in neural network model

development. A good training algorithm can speed up the training procedure with better accuracy.

The final step is the model testing stage, an independent set of input-output samples, called the testing data is used to test the accuracy of the neural model. In case where a neural network model does not meet the requirement an appropriate action need to be done. We could generate more data, select a different neural network structure or use different optimization algorithm. Sometime all the steps listed above need to be done over and over again in order to achieve an accurate model Figure 2.3 is a flowchart of sequential steps involved in neural network model development.

Figure 2.3 Flow charts of the conventional steps involved in neural network model development.

2.3 Conclusions

In this chapter, the step by step neural network based modeling techniques applied to develop an embedded passive component have been presented. The current practice for the neural network model development of a given embedded passive structure involves several steps such as careful selection number of training data, neural network structure, and training algorithm. It often becomes very difficult for microwave engineers to decide the various aspects of model generation, such as the selection of range of input parameters, the selection of neural network structures, and the number of data samples needed for developing a neural model with desired accuracy. While too many samples are CPU intensive, too few samples lead to overlearning of the neural network. Too many hidden neurons need more CPU time and too few neurons result in underlearning of the neural network. Modeling is just one aspect of microwave CAD and the designer wishes to develop neural models even though they do not have detailed exposure to neural network issue. There is a definite need for automation of the neural model development process. In the next chapter we will cover the automatic process to develop the embedded passive component model to be used in multilayer printed circuit board.

Chapter 3

Automatic Neural Network Based Modeling Techniques Applied to Embedded Passive Component Model

3.1 Introduction

The universal approximation theorem [57] forms a theoretical basis for employing MLP neural network to approximate RF/microwave EM behavior with a good accuracy. However, this theorem does not specify what should be the optimal number of training data or the size of the MLP network to provide sufficient generalization capability. Determining appropriate training data and neural network size for model training are the most difficult parts of neural network model development.

Conventional methods to develop reliable neural models have used trial and error techniques to find suitable neural network structure and training data size. This is an expensive method in both human and CPU time for repetitively trying different neural network structures and continuously generating data, and may not generate an optimal solution for a given problem. Initially the randomly selected neural network structure was trained to give a best fitting response to measured or simulated data. If the neural

network model did not produce satisfactory results after extensive training, additional steps were required. The designer might have to add more training data, select different neural network structure or adjust the algorithm parameters. As we see, this conventional method has many disadvantages. One of the disadvantages is that, this is a very tedious and time consuming process and is heavily dependent on the designer's experience. Second, this method wastes a lot of computer time and human effort trying various structures and generates unnecessary data samples. Moreover, this method depends on luck than engineering. Therefore, there is a definite need for automation of neural model development process.

The purpose of automating the neural network model development process is to minimize the human-effort involved and CPU time for facilitating end-to-end ANN-based CAD. This thesis utilizes an automatic model generation (AMG) algorithm [12][58][59] to develop EM-based neural network models for embedded passive components. The conventional method for training a neural model involves several sub-tasks such as data generation, data preprocessing, neural network training, and validation [6]. These sub-tasks have usually been carried out manually in a sequential manner independent of one another. Conversely, the AMG combined with a proposed automatic multilayer selection (AMS) algorithm integrates all subtasks described above into one unified task. The integrated process is computerized and carried out automatically in a stage-wise fashion. Applying both algorithms to model development and data generation from EM solver will reduce manual labor, time for data generation, and any possible chances of human error.

3.2 Automatic Neural Network Based Modeling Algorithms

3.2.1 Automatic Data Generation

The first step in neural network development is to obtain the training data. The training data can be obtained from measured/simulated microwave data. Typical example of simulators that available in the market are *Sonnet-Lite*, *Ansoft-HFSS*, *HSPICE*, *Agilent-ADS*, etc. In the conventional method, obtaining data for neural network model training is highly tedious and error prone especially when extensive data is generated. The reason is that after every simulation, the physical/geometrical or the electrical parameters have to be manually changed to obtain output for the new set of parameters. Furthermore, the total number of samples needed to train a model is unknown. Generating too much data will requires a lot of human and CPU time, and generating less data will lead to an unreliable neural network model.

Having the automatic data generator (ADG) included in the overall automatic model generation has many advantages. The main advantage is to reduce the manual labor and thereby reduce the time for data generation and chances of human error. Unlike the conventional training method where the total number of training data is unknown, by using the AMG (Automatic Model Generation) algorithm the total number of training data can be determined. The AMG is created for automating the neural-network-based RF/microwave model development process. The AMG algorithm utilizes automatic sampling algorithm described in [12] to automate the data generation process. Initially the model is trained with a zero amount of training data and then proceeds in a stage-wise manner to generate the neural network model. During the neural network model training the AMG algorithm dynamically drives the simulation tools to generate

enough data for training and testing. Furthermore, AMG is able to detect non-smooth or smooth sub-regions of the input-output space and then send a request to ADG to generate more or less samples in the non-smooth or smooth sub-regions, respectively. This will avoid generating unnecessary samples and helps in reducing the CPU time significantly. The next section will discuss the automatic driving of the simulation software used to generate data for our examples.

3.2.2 The Key Aspects of Automatic Data Generation

During neural network model development, AMG will send a request to ADG to generate more training and validation data. Once ADG receives the request it will modify the design file with new geometrical/physical, or material parameters information as provided by the AMG algorithm. The design file created by the EM solver software contains all geometrical/physical, material, boundary condition, physical location, and frequency sweep parameters. After the design file is updated with the new information, a new set of output responses will be obtained from the simulator. The results are then converted to the format which AMG can understand. The new data will then be added to the training and validation files. With the new samples added to training data file, AMG will continue to train the neural network further to improve the accuracy of the model. Without the ADG algorithm, the user has to manually run the EM solver, change the dimension structure, wait until solver is completed and then repeat the entire process for all sets of input combinations. This process becomes highly tedious and error prone when extensive data is repeatedly generated. Therefore ADG is one of the key aspects which contributes towards the overall automatic model generation. Figure 3.1 shows the main idea of the ADG algorithm.

Figure 3.1 Systematic framework of the automatic driving data generator algorithm in the forms of a flow-chart.

3.2.3 Verification of Automatic Data Generation Algorithm

In order to verify the AMG and ADG algorithms are working properly *NeuroModelers* [11] and *Sonnet-Lite* [37] are used. *NeuroModeler* was developed with a capability of automatically driving the EM solver to generate data and train a neural network model. *Sonnet-Lite* was chosen as an EM data generator because of its ease of use and speed. For that reason throughout of this thesis we will be using *Sonnet-Lite* as a data generator and *NeuroModeler* for driving data generator and neural network model training.

The first step in ADG is to create the design file using *Sonnet-Lite*. Using this tool we are able to draw the structure of an embedded passive component, define the units of measurement (e.g. mils, mm etc), electrical/physical nature of the materials to be used (e.g. lossless metal, lossy metal, resistivity, permittivity, height of structure, etc) and the boundary conditions (e.g number of ports, location of ports, etc). The version 9 of *Sonnet-Lite* allows us to sweep any geometrical or electrical parameters which helps to significantly reduce the time to setup ADG. We also need to define our desired EM outputs like the S-parameter, Z-parameter or Y-parameter. Once the geometry of the structure and all the parameters are defined, it will be saved as a “*.son” file. The “*.son” file for embedded passive is divided into many sections where we can manually change the physical size of the device.

The next step is to setup *NeuroModeler* for ADG. During the ADG, *NeuroModeler* will search for a key word in the “*.son” file, and find the parameter defined by the user. The value at that location will then be updated with the value passed from AMG. Once all the values are updated to reflect a new change, the design file is resaved and the *em* command of *Sonnet-Lite* will be

executed by the *NeuroModeler* to generate a new set of output responses. When the *Sonnet-Lite* completely solved the new structure and generated the output in the form of S, Y, or Z parameters as specified in the “*.son” file, it generates a separate file known as the “*.cvr” file where all the simulation results are stored. *NeuroModeler* will then need to extract the data stored in the “*.cvr” file and convert it to the format that can be used in model training.

To illustrate the idea let us assume the following example “*Capacitor.son*” file where we want to change the length parameter of the embedded capacitor:

```
** Physical Parameter**
BOX 1 64 48 128 96 20 0
:
:
** Physical Location **
NUM 1
0 5 -1 N 9 1 1 100 100 0 0 0
:
:
**Parameter sweep**
VARSWP
SWEEP 1.0 10.0 5.0
ABS 0.2 20.0
Y
Len_cap y 80 80 80
:
**Output results**
FILEOUT
CSV D Y data_out.csv 15 S RI R 50.00000
END FILEOUT
```

The length of the capacitor is defined in the “Parameter sweep” section. If we look at the 5th line after “Parameter sweep” we notice that it has 5 columns. The first column is a variable parameter “Len_cap”, the next one “y” means the parameter Len_cap will be swept. The next 3 columns are the swept parameters from minimum length to maximum length and the step size between the minimum and maximum. At the present, the length of the capacitor is sweeping from 80 to 80 with step size 80. Assuming that during the neural network training, the AMG algorithm needs to collect data at length 90, first AMG will send the new length value to ADG and request for more data at that point. Then, ADG will search for the keyword “Len_cap” located inside the “*capacitor.son*” file. Once this keyword is found, the second step is to jump to the location storing the minimum, maximum and step size values and have those values changed to 90. After all the values are updated, the “*capacitor.son*” file is resaved and ADG will activate the *em* command of *Sonnet-Lite* to solve the EM responses of the new structure and save the output to the “*data_out.csv*” file. The ADG will then convert the data from “*data_out.csv*” file and import it to the training or validation data file. When everything is finished, the ADG will send a feedback to AMG to continue training. This ADG process will be repeated until the sufficient number of data is needed for model training.

It is clear that having ADG employed with neural network model training significantly improve the neural network model generation. It also will simplify the neural network model generation by reducing the intensive human effort demanded by the step-by-step neural modeling approach. Through this thesis we are going to use the ADG engine to automatically generate data for neural network model training.

3.3 Proposed Automatic Multilayer Selection

3.3.1 Introduction

The next difficulty in neural network model development is to find an optimal network structure for a given microwave problem. The automatic design for artificial neural network (ANN) has attracted immense interest because the performance of ANN depends on its network structure. The performance of ANN for a given problem has been evaluated by generalization ability and the convergence of training time [6]. In many applications, generalization and convergence are non trivial; improving one at the expense of the other becomes a crucial decision. Thus, algorithms that can automatically design near optimal ANN architectures considering both performance parameters are highly desirable.

There have been many attempts in designing ANN automatically such as various constructive algorithms [60]. Roughly speaking, a constructive algorithm starts with minimal network (i.e., a network with minimal number of layers, hidden nodes and connections) and adds new layers, nodes, and connections as necessary during training. The most well known constructive algorithms are the dynamic node creation (DNC) [61] and cascade correlation (CC) [62]. The DNC algorithm constructs single hidden layered ANN architecture with sufficient number of nodes in the hidden layer. However, such single hidden layered architectures have difficulty in learning complex problems. In contrast, the CC algorithm constructs multiple hidden layered structure with one node in each layer. Although the architectural arrangement of the CC algorithm is suitable for

some complex problems for the development of high-order feature detectors [63], it has many practical problems [64]-[66]. In addition, the generalization ability of the network may be degraded when the number of hidden node N is large, because N^{th} hidden node may have some irrelevant connections to predict the correct output. The common problem for both the DNC and the CC algorithms is that there is no way to delete a single connection.

In this thesis we propose a new algorithm; Automatic multilayer selection (AMS), for designing a multiple hidden layered neural network architecture with weight learning. AMS allows each hidden layer to contain more than one node which is automatically determined by the algorithm. It starts the network building process in a constructive fashion by adding more weights to the current training structure. In an attempt to determine the optimal structure quickly, and at the same time avoiding local minima, a linear weight adaptation based on Newton-Raphson extrapolation method is proposed. The Newton-Raphson method is based on the error criteria to dynamically add more weight to current network structure until the learning process is completed. Because the total weights added to the current network structure could become large when using linear weight approximation, a reduction phase is introduced to reduce network size. Once again, to quickly find the optimal network structure after the current network converge, a binary search algorithm is used. The binary search process is repeated until the optimal network structure is found.

3.3.2 The AMS Algorithm

The neural network size, defined as the number of internal weights (freedom), is not known in advance for a given microwave modeling problem. Since it varies for different number of hidden neurons and different neural network structures, the precise network size is not easy to determine. Selecting a large network size may lead to over-learning and require more CPU time for training, while selecting a small network size may lead to under-learning [67]. Highly nonlinear problems require more internal weights while smoother problems require fewer internal weights.

In order to determine an appropriate network size (number of internal weights), the proposed AMS algorithm uses an adaptive process to automatically adjust the network structure and size to best fit the given model problem during the learning process. For a MLP neural network, the total number of internal weights W can be formulated as,

$$W = \sum_{i=1}^{L-1} (1 + n_i) n_{i+1} \quad (3.1)$$

where L is the number of layers and n_i is the number of neurons in the layer i . Generally, the learning capability of a neural network depends on the number of weights (freedom) and structure. The proposed AMS method exploits this concept in the training process to automatically determine the appropriate network structure and network size.

At the beginning, the AMS algorithm constructs all the possible MLP network structures using a default number of hidden neurons. Once the structures are created and sorted

corresponding to the number of internal weights, the proposed AMS algorithm starts the training process.

First, a model with the smallest size will be chosen for optimization. If the desired accuracy is not achieved, the AMS automatically selects the next possible structure and continues the training of that model. If none of the models in the initial candidate set satisfies the desired accuracy, the AMS will automatically create a new set of neural network models corresponding to more internal weights than those in the previous stage. Sorted in order, the new set of neural models will be used in the next AMS training process. This process will be continued until a network model satisfies the desired accuracy.

To illustrate the proposed algorithm, suppose a model with 2 inputs and 2 outputs is to be developed for a given microwave problem. At the beginning, two possible network structures A and B are created corresponding to a default number of hidden neurons (i.e., 2). The structure A is a four-layered Multilayer Perceptrons (MLP4) with one hidden neuron in each hidden layer, and B is a MLP3 with two hidden neurons in the hidden layer as shown in Figure 3.2.

The total number of weights for A and B are 9 and 12, respectively. Initially AMS will select structure A to begin the training process. If the desired accuracy is not achieved, the next structure B is automatically selected for training. Suppose the structure B is still not able to learn the original problem, then a new set of neural network models will be

Figure 3.2 The total number of weights is sorted in order from smallest to largest. A model with smallest size will be chosen for neural network training. If the desired accuracy is not achieved the AMS automatically selects the next larger structure and continue training of that model.

created corresponding to more internal weights. This can be achieved by adding another hidden neurons. All possible network topologies with more neurons (C, D, E, F) are then generated and sorted in order according to the number of weights. Since the structure C has less internal weights than structure B, it is automatically discarded by the AMS algorithm. Structure D, E and F will form the new set of model candidates to be selective for model training. This process will be repeated until a network model with desired accuracy is obtained.

3.3.3 Discussion

3.3.3.1 Minimize the Possible Training Structures

The AMS algorithm trains the model using the default number of hidden neurons. With the number of hidden neurons given, all the possible structures are created and the total number of weights for each structure are then calculated and sorted according to the total number of weight. The AMS trains a model starting with a minimum number of weights and it adds more weights if needed during training to minimize the training error. As number of hidden neurons or number of layers increase the number of possible structures will also increase. For instance, the example shown in Figure 3.2 starts with 2 hidden neurons and a total number of two structures. When one more hidden neuron is added to the current network structure, the number of possible structure is increased to four. As we can see, the number of possible structures grows much faster than the number of hidden neurons added to the model. Therefore the critical part of AMS algorithm's operation is the selection of the structure to be kept and discarded before model training so that the number of possible structures is decreased to minimum.

For the set of all the possible network structures of a given number of hidden neurons some structures would reduce the training error and some could increase the training error. Even with correct architecture parameters, problems of correct choice of initial weights for a certain structure remain unsolved. As mentioned earlier, there is no direct approach in neural network training to give best match with the current modeling problem. Hence, most of the neural network model training is done using trial and error approaches. Most of neural network structure selection is based on experimental results

as described in [68][69]. After a number of trials using different network structures for various problems we are able to reach conclusions for selecting network structures. In our AMS algorithm, in order to select a good candidate set of structures to be used in model training, we will test all the possible structures for various microwave problems. From the results, we are able to draw conclusions on which structures we keep and which we should eliminate. In this thesis, we are using microwave models of three embedded passives namely resistors, capacitors and spiral inductors. These models are highly non-linear and each has its own unique characteristics independent to each other, making it is suitable to learn the neural network model behavior for different neural network architectures.

The embedded passive models have a set as follow: The neural network model of an embedded resistor having an input vector containing the resistor length (L : 6 to 30 mils), Width (W : 6 to 30 mils) and signal frequency (f : 1 to 20 GHz), will have an output of a neural network containing the real and imaginary parts of S-parameters RS_{11} , IS_{11} , RS_{21} , and IS_{21} . The setup for an embedded square capacitor is the same as resistor except for the inputs to the neural network which are a capacitor of length (L : 6 to 20 mils), dielectric of thickness (t : 0.2-0.6 mils), and signal frequency (f : 1 to 20 GHz). The setup for a spiral inductor is a little bit different. The inputs of a neural network model for spiral inductor consist of the length (W : 8 to 10 mils), separation (s : 2 to 4 mils) and the frequency (f : 1 to 20 GHz). A fast 2.5-EM simulator *Sonnet-Lite* is used as a data generator for model training. Figure 3.3 shows three embedded model developed using *Sonnet-lite*. We utilize the automatic data generation as discussed in section 3.2 to generate the data for neural network model training.

Figure 3.3 3D and 2D layout of the embedded components created using *Sonnet-Lite*.

(a) An embedded resistor, (b) An embedded capacitor and (c) A spiral inductor.

3.3.3.1.1 Choice of the Number of Layers

Once the data for the three embedded passive devices are obtained, the next step is to devise a strategy to minimize the number of possible neural network structures for a given number of hidden neurons. The first target is to minimize the number of layers. Adding more layers to network structure would increase many possible structures. This gives rise to the following question: How many hidden layers are required to classify P patterns using MLP structure? The universal approximation theorem [57] which guarantees that one layer network can model any behavior if it has an appropriate number of hidden neurons. Whereas using only one hidden layer could make the network size ridiculously large. In order to reduce this complexity multilayer neural networks are used. For certain cases two hidden layers are proven to be more efficient to use than single hidden layer network [69][70]. Three or more hidden layers are rarely used because the networks are more prone to fall into bad local minima [68][71]. Since the precise number of hidden layers required for a given microwave-modeling problem remains an open question [72], we should investigate the effect of using one, two or more hidden layers. One of the parameters that influence the generalization capability of a network is the complexity of the network [73]. We thus need to compare three and four layer networks of the same complexity. Baum and Haussler [73] have shown that the complexity of a model can be approximated by the number of weights in the neural network architecture. Since the exact complexity of the model is practically impossible to calculate, therefore the number of weights will be used as an approximate indicator of complexity. This means that we can compare different network architecture with the same number of weights.

In order to learn how the network architecture is going effect neural network training, a set of different neural network topologies consisting of MLP3, MLP4, MLP5 are developed. In this experiment we create all the possible neural network topologies with the same number of total weight. To make the model training consistent, each of the network structure is trained under the same conditions, the same total number of training, validation samples and training algorithm on the same computer. The total number of training data is 1200 samples and validation data is 1500 samples for all three embedded passive models. The internal weights of each structure are randomly initialized by the neural network model training algorithm. Table 3.1 shows the neural network structures for different number of hidden neurons but having the same total weights.

Table 3.1 Different neural network topologies having same total weight

Number of Hidden Neuron	Hidden Neuron in Layer 1	Hidden Neuron in Layer 2	Hidden Neuron in Layer 3	Total Weight
5	5	0	0	44
7	5	2	0	44
7	1	6	0	44
8	7	1	0	44
8	4	2	2	44
8	3	3	2	44
8	3	1	4	44

The results obtained from training the three neural network models for embedded passive structures are shown in Figure 3.4. As we compare the three different network structures MLP3 (3-5-4), MLP4 (3-5-2-4) (representing 3 inputs, 5 hidden neurons in first layer, 2 in second layer and 4 outputs), and MLP5 (3-4-2-2-4), the MLP3 seems to be out performs the other two. The training errors are different for all three models but they still share something in common. For small number of hidden neurons, the MLP3 seems to

Figure 3.4 Comparison of percentage training error versus different neural network topology of MLP3, MLP4, and MLP5. (a) An embedded resistor, (b) An embedded capacitor, (c) An embedded inductor.

be out performing the MLP4 and MLP5. Hence, at the beginning when the neural network structure is small only the MLP3 is used and the MLP4 and MLP5 are constructed as more hidden neurons are added to the current neural network structure.

3.3.3.1.2 Minimize the Neural Network Topologies

Minimizing the number of layers helps to reduce the number of possible structures for training. Assume that we only use MLP3 and MLP4 for neural network model having 3 inputs and 4 outputs, and let the number of hidden neurons to be 9. The total number of possible structures for both MLP3 and MLP4 is 9. During training if one hidden neuron is added to the current network structure 10 additional more structures are possible. As more and more hidden neurons are added to the current structure, the number of possible structures will grow very large and it is not possible to try them all. For this reason, the next objective in structure minimization is to investigate which network topology would help to minimize the training error and which non efficient structures can be eliminated at the beginning to reduce the training time. For instance, let us compare two structures having the same weights and hidden neurons such as 3-5-4-4 and 3-2-7-4. Which network topology would give better model training? If we have experienced with neural network training we will notice right away that the model trained using 3-5-4-4 has smaller error compare to 3-2-7-4. For the 3-2-7-4 structure there are only 2 hidden neurons in first layer but 7 hidden neurons in the second layer. So the neural network freedom is limited by the few hidden neurons at the input, and too much freedom at the output. Too much freedom at the output leads to overlearning and poor generalization ability [67]. Training of neural network and having such a configuration, easily leads to a

local minimum. To study the effect of the number of neurons in the hidden layer during training, we train all the network structures with different network topologies. From the training results we are able to make a decision as to which network topology to keep and which one to discard. The data used to train the various network topologies are obtained from the three embedded passive models above. To make the comparison as fair as possible, all network structures with different network topologies are trained using the same conditions i.e., such as the number of training, validation data, and using the same computer. The results obtained from three embedded passive components are shown in Table 3.2. For easier comparison the results are sorted according to the training error.

The results from Table 3.2 give us a better understanding about the performance of each neural network structure topology. Observing the results, we notice that the structure having a large difference in the number of hidden neurons between two layers gives a large training error, and the network topology having more hidden neurons at the output gives a smaller training error than the network having more hidden neurons at the input. The experimental results make sense compared to the actual analysis of the network topology. Having large differences in the number of hidden neurons between consecutive hidden layers will minimize the learning capability and storage capacity of the network [74]. The results we obtained agree with the sensitivity analysis of multilayer perceptron [71] and the neural network topology analysis [68].

Table 3.2 Comparison training error of different neural network topologies for MLP4 (a)

An embedded resistor, (b) An embedded capacitor, (c) An embedded inductor

(a)

Number of Hidden Neuron	Total Number of Weight	Neural Network Structure	Percentage Training Error for Resistor Model
9	65	4-5	0.093
9	64	3-6	0.131
8	56	3-5	0.15
9	64	5-4	0.15
7	48	3-4	0.39
9	61	6-3	0.77
8	56	4-4	0.78
7	47	4-3	0.82
7	47	2-5	2.23
8	54	2-6	2.52
9	61	2-7	2.62
7	44	5-2	2.71
8	50	6-2	2.71
9	56	7-2	2.71
8	54	5-3	2.72
8	50	1-7	3.43
7	44	1-6	3.52
9	56	1-8	3.52
8	44	7-1	4.17
9	49	8-1	4.18

(b)

Number of Hidden Neuron	Total Number of Weight	Neural Network Structure	Percentage Training Error for Capacitor Model
9	65	4-5	0.6
8	56	3-5	0.82
9	64	3-6	1.13
8	56	4-4	1.21
7	48	3-4	1.32
9	61	6-3	1.35
9	64	5-4	1.55
7	47	4-3	2.06
7	47	2-5	2.6
8	54	2-6	3.55
8	54	5-3	3.75
9	56	7-2	3.8
9	61	2-7	3.8
8	50	6-2	4.35
9	56	1-8	4.75
8	44	7-1	5.56
8	50	1-7	5.75
7	44	5-2	6.41
7	44	1-6	6.5
9	49	8-1	7.5

(c)

Number of Hidden Neuron	Total Number of Weight	Neural Network Structure	Percentage Training Error for Inductor Model
9	65	4-5	9.29
8	56	3-5	12.67
9	64	3-6	12.73
8	56	4-4	13.51
7	48	3-4	13.79
9	61	6-3	14.16
9	64	5-4	15.11
7	47	4-3	15.45
7	47	2-5	15.6
8	54	2-6	16.05
8	54	5-3	16.75
9	56	7-2	17.01
9	61	2-7	18.48
8	50	6-2	18.5
9	56	1-8	20.02
8	44	7-1	20.04
8	50	1-7	20.21
7	44	5-2	20.46
7	44	1-6	20.58
9	49	8-1	20.73

The preliminary results of our experiments and the consequent analysis [68][71], do disclose significant guidelines about the appropriate number of layers in a network and the appropriate number of neurons in each layer and other information that could be obtained prior to the network design and implementation. One or two number of layers is generally enough in the beginning. Adding more layers will cause more difficulty in training and make it prone to falling in local minima during training. From the experiments we have also found that four layer networks are easier to train if the numbers of neurons in the hidden layers are balanced. This led us to believe that the training algorithm is more likely to get stuck in a local minimum if the numbers of neurons in the hidden layers are not balanced. In terms of number of hidden neuron per layer, we observed that the closer a layer to the output layer was the more effect the number of

neurons in the layer had. The above points led us to conclude that there seems to be no reason to use more than 2 hidden layers. For the network topology selection we only select the network structure having roughly the same hidden number of neurons for each layer, and a hidden layer close to output layer will always have more or the same number of hidden neurons compare to the first layer.

Applying those two neural network structure criteria the number of possible structures is significantly reduced. For instance, let assume a model with 3 inputs, 4 outputs, and 3 possible hidden layers with the number of hidden neurons ranging from 5-9 hidden neurons. The number of possible structures is 115. Assuming that the training time for each structure is 20 minutes, for 115 structures the total training time will be 39 hours. By applying the neural network structure reduction criteria, the number of possible structures will be reduced from 115 down to 10 and the total time to train 10 structures will be reduced to 3.3 hours.

3.3.3.2 Speeding Up the Neural Network Model Training Using Newton-Raphson Method

After applying network layer and network topology minimization we eliminate all the non-efficient structures from tables Table 3.2 to form a final Table 3.3. We observe that as more weights are added to the current network structure the training error is decreased. To have a better understanding how the total weights added to the network structure can improve neural network training we draw graphs of total weights versus the training error for three embedded passive models of a resistor, capacitor and inductor.

Table 3.3 Training errors sorted after eliminating non-efficient structures for three embedded passives models

Total Hidden Neuron	Total Weight	Neural Network Structure	Percentage Training Error		
			Resistor	Capacitor	Inductor
5	33	2-3 *	5.23	3.23	16.84
6	40	2-4	2.12	2.46	15.97
5	44	5	0.5	2.3	15.75
7	48	3-4	0.39	2.2	13.79
6	52	6	0.168	1.99	13.52
8	56	3-5	0.15	1.58	12.67
7	60	7	0.116	1.32	9.45
9	65	4-5	0.093	0.92	9.29
8	68	8	0.08	0.85	8.27
9	76	9	0.061	0.68	7.01
* 2-3 means 2 hidden neurons on first layer and 3 neurons on third layer					

Observing the results of the 3 graphs of Figure 3.5 on the next page we discover that the neural network model training error is decreasing as total weights increase. The error function curve can be approximated by a linear or quadratic function. By using Newton-Raphson extrapolation method [75] we are able to calculate the approximate weights to add to a current network structure to be able to achieve a desired training error more quickly.

(a)

(b)

(c)

Figure 3.5 Comparison the total weight versus the training error after eliminating the non-efficient structures. (a) An embedded resistor, (b) An embedded capacitor, and (c) An embedded inductor.

Newton-Raphson uses ideas from calculus to find the zeros of an arbitrary equation. The underlying idea is the approximation of the function $f(x)$ by the derivative of $f'(x)$ with an arbitrary x . The Newton-Raphson formula consists of geometrically extending the tangent line at the current X_i , until it crosses zero, then setting the next guess X_{i+1} to the abscissa of that zero-crossing. Figure 3.6 shows the idea of Newton's method extrapolates the local derivative to find the next estimate of the root.

Let r be a root (also called a "zero") of $f(x)$, that is $f(r)=0$. Assume that $f'(r) \neq 0$. Let x_1 be a number close to r (which may be obtained by looking at the graph of $f(x)$). The tangent line to the graph of $f(x)$ at $(x_1, f(x_1))$ has x_2 as its x -intercept.

Figure 3.6 Applying Newton-Raphson extrapolation method to calculate the approximate weights for the next neural network model training.

To find a formula for x_2 in terms of x_1 we use the fact that the slope of the tangent line is $f'(x_1)$, so its equation is

$$y - f(x_1) = f'(x_1)(x - x_1) \quad (3.2)$$

since the x-intercept of the tangent line is x_2 , we set $y = 0$ and obtain

$$0 - f(x_1) = f'(x_1)(x - x_1) \quad (3.3)$$

if $f'(x_1) \neq 0$, we can solve this equation for x_2 :

$$x_2 = x_1 - \frac{f(x_1)}{f'(x_1)} \quad (3.4)$$

Assuming $f'(r) \neq 0$, we will not have problems with the denominator being equal to 0.

We continue this process and find x_3 through the equation:

$$x_3 = x_2 - \frac{f(x_2)}{f'(x_2)} \quad (3.5)$$

This process will generate a sequence of $\{x_n\}$ which will approximate r . To apply Newton's method extrapolation for neural network weight prediction we replace the parameter x by the total number of weights and the value of y by the training error. Using Equation 3.1, we are able to calculate the total weight of the current network structure. The normalized training error of the current network structure at the end of k th stage is defined as

$$E_l^k = \frac{1}{N_l^k} \sum_{x_q \in L^k} (e_q(w^k))^p \quad (3.6)$$

where p represents the p^{th} norm, N_l^k is the number of samples in L_l^k , and E_l^k is the error due to q^{th} sample in L given by

$$e_q(w^k) = e(x_q, w^k) = \frac{1}{N_y} \left[\frac{1}{P} \sum_{j=1}^{N_y} \left| \frac{\tilde{y}_j(x_q, w^k) - y_{qj}}{y_{\max,j} - y_{\min,j}} \right|^p \right]^{\frac{1}{p}} \quad (3.7)$$

where $\tilde{y}_j(x_q, w^k)$ is the j^{th} neural-network output at the end of the k^{th} stage training for input x_q , and y_{qj} is the j^{th} element of y_q . The $y_{\min,j}$ and $y_{\max,j}$ are the minimum and maximum possible values of y_j data in the input space of interest.

Now let the training error of the previous stage be $\hat{E}_T^{k-1}$, the total weight of the neural network structure at the previous stage be W_T^{k-1} , and the desired training error be E_d . The training error of the current stage be $\hat{E}_T^k$ and total weight corresponding is W_T^k . Once the training errors of two stages are obtained, the next prediction network size W_p can be approximated using Newton's method to quickly get the model training error closer to desired accuracy. To find W_p , we first calculate the rate of change of the training error with respect to the rate of change of the total weight added to the network using.

$$\eta = \frac{\Delta \hat{E}_T^k}{\Delta W_T^k} = \frac{\hat{E}_T^k - \hat{E}_T^{k-1}}{W_T^k - W_T^{k-1}} \quad (3.8)$$

If the rate of change of training error with respect to the total weight added to the network is not zero, then we can replace the x_2 of Equation 3.4 with the next prediction weight W_p . The formula to calculate the next prediction weight is

$$W_p = W_T^k + \frac{E_d - \hat{E}_T^k}{\eta} \quad (3.9)$$

Once W_p is obtained, the next MLP structure W_T^{k+1} will be calculated to have the total weight closely match W_p . The neural network model will continue training with a structure having W_T^{k+1} weights. After training if $\hat{E}_T^{k+1}$ is more than desired accuracy the W_p will be recalculated based on $W_T^k, \hat{E}_T^k$ and $W_T^{k+1}, \hat{E}_T^{k+1}$ to find W_T^{k+2} . This process is repeated until convergence is reached.

During neural network training, sometime the rate of change of training error with respect to the total weight added to the network is too small. The possible remedies are perturbing the current solution for w or adding 10-20% more weight to the current network size to escape from a local minimum of $E_T(w)$, and then continuing the training.

3.3.3.3 Optimize the Network Structure.

The ideal neural network model training algorithm is not only to train a model to meet the user desired accuracy but also to optimize the neural network size to be as small as possible. The advantages of having a small network size are having a faster and easier model to import into circuit simulators since the structure of the model is simpler, and the model can generalize better. Large neural network models tend to memorize the training data but cannot generalize well [6]. After using Newton's method to find the neural network size that can exceed the user desired accuracy, the next step is to reduce the network size. To find the optimal network size for a given microwave problem we are going to use the binary search algorithm [76]. The binary search is used to locate the

optimal network size between the maximum network size determined by the Newton's method and the minimum network size before model convergence.

Let us assume that the maximum network size is defined as W_{right} , the one before solution is found called W_{left} and the optimal network size is somewhere between W_{left} and W_{right} . W_{predic} is used to find an optimal structure somewhere between W_{left} and W_{right} . By using the binary search the W_{predic} is always halfway between positions W_{right} and W_{left} . If the result of neural network training of W_{predic} is less than desired accuracy, the network size of W_{right} will be updated to W_{predic} . If the W_{predic} is more than desired accuracy, then W_{left} will be set as W_{predic} . To find the next optimal network size, W_{predic} will move to the next location halfway between W_{left} and W_{right} . This procedure will be repeated as long as there is a possible integer network size between W_{left} and W_{right} . The optimal network size will be the final value of W_{right} .

3.3.3.4 Combination of Newton-Raphson Method and Binary Search Algorithms

To illustrate both methods of Newton-Raphson and the binary search algorithm used in neural network size selection, we assume the training error for a microwave problem as shown in Figure 3.7.

Neural network training is started with network structure having the total weight of A. Once the training is completed, assuming that the training error is obtained, W_{right} is set to A. The prediction total weight of the next structure B is 10-20% of A and W_{predic} is set

Figure 3.7 Demonstration of using Newton-Raphson algorithm.

to location B. With error obtained from B and combined with the error from A, the Newton's method can now be used to calculate the next prediction total weight. Before doing that we set W_{left} to W_{right} and update the W_{right} with $W_{predict}$. By using Equation 3.8 and 3.9 the next network size is calculated and the next $W_{predict}$ is found to be D. The parameter W_{left} now moves to W_{right} , W_{right} replaces $W_{predict}$ and $W_{predict}$ is moved to location D as shown in Figure 3.7. Since D is still in underlearning, the algorithm continues to use the Newton's method to calculate the next possible structure at F. Again, all the parameters W_{left} , W_{right} , and $W_{predict}$ are updated accordingly. At location F, it yields a solution but it does not guarantee an optimal solution. The next step is to find an optimal network structure which is shown in Figure 3.8.

Figure 3.8 Demonstration of using Binary search Algorithm.

Once a solution is found, we set the W_{right} at the location F and then use the binary search to find an optimal solution. The next structure size is calculated using the following formula

$$W_{predict} = \frac{W_{left} + W_{right}}{2} \quad (3.10)$$

At E, the result of training error is more than desired accuracy but less than W_{left} . therefore, W_{left} will move to location of $W_{predict}$ and the next location is calculated. Since there is no structure available between W_{left} and new W_{right} , the algorithm will stop. The optimal structure will be structure F.

This example demonstrates Newton's method to find convergent solutions more quickly and binary search to help find the optimal structure. The linear network size adaptation requires six iterations to find an optimal solution. On another hand by using the

combination of the Newton's and binary search techniques it only requires five tries to find an optimal solution. As the model gets more complex and a larger neural network size is required the proposed method becomes more effective. Chapter 4 shows the strength of this algorithm when training the complicated structure such as a spiral inductor.

3.3.3.5 Converting the Prediction of Total Weight to an Appropriate Network Structure

The prediction of total weight (W_{predic}) obtained using Newton's method or Binary search cannot be directly converted to an MLP structure. Therefore the MLP network structure having a total weight close to the prediction weight will be selected. To calculate the network size having the total weight close to the prediction weight for the MLP3 structures the Equation 3.1 needs to be reversed. Instead of calculating the total weight we need to calculate the total number of hidden neurons. The reversed formula is

$$H = \frac{W_p - y_i}{1 + x_i + y_i} \quad (3.11)$$

Where H is the total number of hidden neurons, x_i is the number of inputs and y_i is the number of outputs. The result of H could be a real or integer number. If H happens not to be an integer, the number of hidden neurons will be rounded off to the nearest integer.

For MLP4 and MLP5 the number of hidden neurons for each layer is unknown. To find number of hidden neurons required for each layer, the multivariable formula need to be solved. Another way around is to optimize the hidden neurons for each layer so that the total weight of the network structure is close to the prediction weight. Based on the

constraints of the network topology selection criteria, the optimization hidden neurons per layer for an MLP is simplified. Basically the number of hidden neurons per layer for the MLP structure of a neural network must be about the same between the layers or the total number of hidden neurons closer to the output layer is slightly larger than that in other layers. The algorithm to calculate the number of neuron for each layer is described as follow.

Let us define the total number of hidden neurons as H , the number of hidden neurons at layer i as h_i , and the total number of hidden neurons accumulated for each hidden layer as h_{acc} . L is the total number of layers, W_{predic} is prediction weight and W is the total number of weights for the current structure.

Initialize the default structure having H, L . We then calculate W using Equation 3.1

While $W < W_{predic}$

$H = H + 1$; // Increase H by 1

For index from 1 to $L-2$ //number of hidden neuron for each layer should be equal

$h_{index+1} = \text{Truncate}(H / L)$ //round off number of hidden neuron to nearest integer

$h_{acc} = h_{acc} + h_{index+1}$ //Calculate accumulate number of hidden neuron

$h_{L-1} = H - h_{acc}$ (hidden layer close to output will have more or equal to other hidden layer)

Calculate total weight W of the new structure using Equation 3.1

When the while loop is completed with the given number of input(s) and output(s), the next training structure with the hidden neurons for each layer will be determined by the algorithms described above.

3.4 Overall Automation of Neural Network Model Generation

3.4.1 Combination AMG, ADG and AMS Algorithms

AMG [12][58] combined with the proposed AMS are designed to shift the workload from human to computer thus making model development more efficient and less prone to errors. In the manual approach the most difficult parts in neural model development are data generation, neural network structure selection, and neural network training. By combining the AMG, ADG, and AMS algorithms together, all the subtasks described above will be integrated into one unified task. The integrated process is computerized and carried out automatically in a stage-wise fashion. The overall AMG algorithm included systematic driving of automatic data generation, neural network training and simultaneous model testing. During network training the AMS algorithm helps to determine neural network size based on neural network training error. Combining these three algorithms improves the speed of the model generation considerably.

The combined AMG, ADG and AMS algorithms utilize stage-wise training and adaptive data sampling to automate neural model generation. Initially, neural network with relatively few hidden neurons are trained with training samples from the available data. The training data could be empty at the beginning, and during the training process AMG automatically drives ADG to add more samples to the training data. The resulting neural model is validated with data samples automatically generated during the process. The algorithm stops if the error is less than or equal to the error specified by the user. Otherwise, the algorithm automatically takes suitable actions and proceeds to the appropriate stage. These stages are as follows:

1. Automatic sampling and generation of training and validation data: Over-learning of neural network may be detected at the end of the k^{th} stage, using error information of the training and validation. Over-learning is a phenomenon in which the neural network memorizes the training data accurately but cannot generalize well. That happens when validation error is greater than training error. When the algorithm detects over-learning, it dynamically adds more data samples to the training and validation sets.

2. Automatic neural network structure adaptation: Under-learning of the neural network may be detected after a certain stage, using training error and its gradient error. Under-learning is a phenomenon in which the neural network has difficulties in learning the training data itself. That is when training error is large and gradient error is small. When the AMG algorithm detects under-learning, it sends a request to AMS to expand the neural network size. A larger neural network size will then provide increased freedom to better learn the non-linearity in data training. Based on the neural network size and training error AMS is able to calculate an appropriate neural network size for the next training.

3. Automatic detection of local minimum in neural network training: After going through a number of trials and recognizing that model accuracy is not improving, AMG will randomly perturb network weight w and go to automatic training.

4. Overall Automation: At the end of each stage, the algorithm checks for various possible neural network training situations and takes relevant actions, such as updating data, adjusting neural network size, and etc.

3.4.2 Details Expanding of Overall Automatic Neural Network Model Generation

In general, RF/microwave component and circuit behaviors are nonlinear and multi-dimensional, and may be composed of a variety of patterns such as smooth, non-smooth, low-frequency, high frequency etc. This makes it difficult to formulate a general equation to generalize all the microwave behavior. In most of the cases, neural network model training starts with small neural network size. Based on the training error at the end of each stage, the AMG algorithm will make a decision on how many more weights are needed to current network structure. The AMG algorithm will continue training the model using the network topology passed from the AMS algorithm. The objective of combined algorithms AMG, ADG, and AMS is to conveniently develop a neural network model $y=h(x,w)$ that accurately represents the complicated RF/microwave input/output relationship $y=f(x)$ of interest.

Let $h^k(x,w^k)$ represent neural network structure in k^{th} stage with outputs $y^k=h^k(x,w^k)$, where w^k is the corresponding weight vector. Let E_t^k, E_v^k and ΔE_t^k represent training error, validation error and gradient of the training error of the neural network $h^k(x,w^k)$ at the end of the k^{th} stage respectively. The user sets neural network model accuracy and maximum allowable number of stages k_{max} . In the first stage, training data (T^1) and validation data (V^1) are automatically generated using ADG algorithm. Once the (T^1) and (V^1) data are obtained, the AMG algorithm begins to train a neural network model using a default network structure $y^1=h^1(x,w)$. The objective of neural network training in k^{th} stage is to optimize w^k such that E_t^k is minimized. The normalized training error

Equation 3.6 is used to test the model accuracy for each of the iteration for each stage.

The resulting model training at the end of each stage is validated with data samples V^l .

The normalized validation error of neural model $h^l(x, w)$ is defined as

$$E_v^k = \frac{1}{N_v^k} \sum_{x_q \in V^k} (e_q(w^k))^p \quad (3.12)$$

where N_v^k is the number of samples in V^k , and $e_q(w^k)$ is the error due to the q^{th} sample in V^k , computed using Equation 3.12. At the end of each stage, AMG will check whether k has been reached to k_{max} or the condition $\{E_v^k \leq E_d\}$ has been satisfied. If any of them has been satisfied, AMG algorithm will stop; otherwise it will check the current network training status for overlearning, underlearning or possible fall into local minimum in neural network training.

Overlearning can be detected when $E_t^k < \beta E_d$ (where β is a constant controlled by user) and $E_v^k \gg E_t^k$. The suitable action is taken by adding more samples to (T^k) and (V^k) when overlearning is detected. In the case when samples can not be generated AMG will send a request to AMS to reduce the weight of the current structure. AMS will use the binary search algorithm as described in section 3.3.3.3 to reduce the total weight of the $h^k(x, w^k)$ and pass it to AMG for further training.

Underlearning can be detected when $E_t^k < \alpha E_d$ (where α is a constant controlled by user) and $\Delta E_t^k < \Delta E$. The action taken when underlearning is detected is to add more weight to $h^k(x, w^k)$. In this case AMG will send a request to AMS for a larger network size. The

AMS first checks the $h^k(x, w^k)$ for its convergence. The reason is that if $h^k(x, w^k)$ is convergent, it indicates the next step is to find the optimal network structure. By using the binary search to find an optimal structure, the prediction structure can sometimes be in underlearning stage. Therefore the condition to check $h^k(x, w^k)$ for convergence helps AMS to decide whether to use either Newton's method or binary search for the next weight calculation $h^{k+1}(x, w^{k+1})$.

In the case when neither overlearning nor underlearning is detected, indicates that the current network is possible at a local minimum. When this happens, the AMG algorithm will randomly perturb network weights w^k and then go back to automatic training. The completed algorithm of AMG along with ADG and AMS is shown in Figure 3.9.

Figure 3.9 Systematic framework of the proposed combination of AMG, ADG and AMS algorithms in the form of a flow-chart.

3.5 Conclusions

The algorithms have all been tested and verified during this thesis research. Combined AMG, ADG and AMS techniques eventually leading to a new automatic neural network model training for multiple network layers, which is the first of its kind in the existing EM-ANN literature. Given an RF/microwave ANN-modeling problem $y=f(x)$, user provides the specifications of a desired neural model accuracy, related to the training validation data from the simulator and corresponding high frequency simulation software, among other specifications. The combined AMG, ADG and AMS will then automatically generate a satisfactory neural network model $\hat{y}=h(x,w)$ representing the detailed theoretical relationship $f(x)$.

Chapter 4

Applying Automatic Model Generation for Developing Example of Embedded Passive Component Models

4.1 Introduction

The next generation of high frequency communication products will consist of mixed signal modules with numerous digital and analog RF circuits tightly integrated in a substrate. For these products, a new area of technology is emerging that uses embedded passive components to achieve high reliability, high performance, tight integration, and low cost [4]. As the signal speed increases, the dimensions of embedded passives in multi-layer circuits become a significant fraction of signal wavelength. The conventional electrical models for the components are no longer accurate. Models with physical/geometrical information including electromagnetic (EM) effects become necessary. Furthermore, high-speed circuit designs need statistical analysis, and yield optimization taking into account process variations and manufacturing tolerances. Hence, it is extremely important to develop various values of physical/geometrical dimensions of embedded passive components over a useful range of operating frequencies. The resulting component models including EM effects can be used in RF

circuit simulators to model the performance of a circuit at a range of operating frequencies. During the simulation we are able to observe the impact of EM effects on the circuit, and since the components are continuous in physical/geometrical dimensions, we will be able to optimize them to give the optimal circuit solution. Applying these models in circuit simulation, significantly reduces the number of design iterations required to develop RF modules, and reduces the required time to bring a new product to market.

Presently, the cost of individual neural network models has been made manageable. Massively developing neural network models for varying values of physical/geometrical dimensions require massive data generation and repeated model training. Development of large embedded passive component models could be a highly intensive process when using the conventional neural developing method. By using the advanced AMG method, human tasks are converted into computer based tasks to facilitate 24-hour nonstop work cycles even in the absence of humans, and most importantly removes the element of risk or possibility of human-induced errors. Therefore it would be possible to use the new advanced AMG method to develop a large set of accurate models to be used in frequency/time domain simulators.

4.2 Development of Embedded Resistor Neural Network Model Using Advanced AMG Approach versus Conventional Method

In order to demonstrate the AMG approach, we utilized the pure neural network to develop EM-based neural network models for an embedded resistor using training data from an EM simulator. This example is the same embedded resistor example we used in Chapter 3 for neural network structure analysis. The only differences are that the range

of the inputs to the neural network model are wider, and training/validation data is not available at the beginning. Since the behavior of the embedded resistor is not complex, we are able to use a small neural network size to train this model. The purpose of this example is to compare the performance of a neural network training using the conventional method against the new advanced AMG method. The input parameters include length (L : 6 to 30 mils), width (W : 6 to 30 mils), and frequency (f : 0.1 to 1GHz; step size 0.1GHz and 1 to 20GHz; step size 1GHz). As for the data to train the neural network model, the conventional method is using a constant step size, but for the AMG method, the data is automatically generated by the AMG algorithms. In both methods, we will continue to use *Sonnet-Lite* [37] for the data generation, and *NeuroModeler* [15] for neural network training. The object now is to train a neural network model that will represent the $x - y$ relationship and that matches the generated data.

First we use the conventional step-by-step neural modeling approach to develop the neural network model for the embedded resistor. By using the conventional approach, we are unable to determine which area of inputs space is linear or non-linear so a constant step size is used. At the beginning, during neural network training we use a large step size. As more data are needed, the step size will reduce accordingly. To train the neural network using the conventional method, we start with small network size of 5 hidden neurons. During training, the trained model will test against the training and validation data that is manually generated. If the neural network structure can not match the validation data, more hidden neurons are added by trail and error until the desired model accuracy is achieved. Once the desired accuracy is achieved new validation data is

generated for further model accuracy. If the model does not match the new validation data, the old validation data will be added to the training data and then will be replaced with the new ones. The neural network model will continue to train using the same method mentioned above. This process will be repeated until the new validation data matches the current trained neural network model. Table 4.1 shows the step-by-step approach based on conventional neural network training.

Table 4.1 Neural network model training of the embedded resistors using conventional approach.

Training Stage	Training Sample	No. Hidden Neurons	Model Accuracy %	Decision for Next Stage
1	150	5	25.54	More Neurons
2	150	6	19.25	More samples
3	300	6	12.35	More Neurons
4	300	8	8.78	More samples
5	500	8	4.35	More samples
6	750	8	2.15	More Neurons
7	750	10	1.32	more samples
8	900	10	0.56	Stop

Now the new advanced AMG method is used to train the embedded resistor neural network model. In the first stage of training, a neural network starts with 5 hidden neurons. A total of 6 training and 2 validation samples are used. When the automatic algorithm decides to add more data it dynamically drives the *Sonnet-Lite* simulator. After 6 stages, the final neural model with 8 hidden neurons, 467 training and 230 validation samples, achieves a validation error of 0.15%. To confirm the reliability of the generated neural network model, it is further tested with a large set of independent test data never seen during AMG's model development process. The average test error is observed to be

0.27%, which satisfies the $E_d=1\%$ specifications. The reported neural network training using advanced AMG is shown in Table 4.2. From two methods of neural network training, it can be seen that the advanced AMG gives more accurate models with similar amounts of training data as compared to the step-by-step approach. This is because the advanced AMG algorithm uses an efficient sample distribution, network size selection, and network training algorithm. The result of neural network model training using advanced AMG for the embedded resistor is shown in Figure 4.1

Table 4.2 Neural network model training of the embedded resistors using advanced AMG approach.

Training Stage	Training Sample	No. Hidden Neurons	Model Accuracy %	Decision for next stage
1	132	5	22.32	More Neurons
2	132	7	13.32	More samples
3	320	7	7.12	More samples
4	390	8	3.42	More samples
5	420	8	1.21	More samples
6	467	8	0.15	stop

Figure 4.1 The outputs of the neural network model for the S-parameters of embedded resistors match very well with EM test data which are never used in training process.

We also perform the time comparison between the AMG method and the step-by-step manual neural modeling approach. The comparison result is shown in Table 4.3. It can be seen that the human time required in the case of the AMG method is very small compared to the manual approach. The reason is that the data generation and neural network model training in our method are automatic as opposed to manual data generation in the step-by-step approach. The CPU time of AMG is also smaller since the manual approach requires a relatively larger number of training samples, and optimization to achieve similar model accuracy.

Table 4.3 Comparison of the time taken by the AMG method and manual step-by-step neural modeling approach for the embedded resistors.

Method	Human Time	CPU Time	Total Time
AMG	5 min	24 min	29 min
Manual	300 min	64 min	364 min

4.3 Development of 2-Loop Spiral Inductor Neural Network Model Using Advanced AMG Approach

In this example, we modeled a 2-loop inductor to demonstrate the robustness of our approach. The spiral inductors are essential for bias injection into oscillators, amplifiers and microwave switches. They can also be used to bias tuning varactors, PIN diodes, transistors and monolithic circuits [17]. The parameters of interest for the design of inductors are inductance (L), quality factor (Q) and self resonant frequency (SRF). Unfortunately there are no simple formulas to determine these parameters accurately. Some of them rely on electromagnetic simulation while others use simple lumped

element models. The programs which use electromagnetic simulation are very expensive and often take a few hours to days to run a complete simulation. On the other hand, existing lumped element approaches of first order analysis do not give an accurate value for the critical parameters like Q and SRF [77]. For this reason we are going to use a neural network to model the behavior of the spiral inductor. The result of the model can be used in a circuit simulator.

A 2-loop spiral inductor is modeled using EM-based training data automatically generated using *Sonnet Lite*. This example is the same embedded inductor example we used in Chapter 3 for the structured analysis. The accuracy of the neural network model for the inductor in Chapter 3 was trained using a fixed number of training samples and neural network size to achieve had an accuracy of 9.26%. In this chapter we will continue to train a model to achieve a model accuracy of 1%. The neural model of the 2-loop spiral inductor is shown in Figure 3.3. Input x contains width (W : 8 to 10 mils), space (s : 2 to 4 mils), and frequency (f : 1 to 20 GHz). Real and imaginary parts of S-parameters RS_{11} , IS_{11} , RS_{21} , IS_{21} , RS_{22} , IS_{22} are the model outputs y . The conventional step-by-step neural modeling development for this model involved the following steps:

1. Graphically re-drawing the inductor with new physical parameter values
2. Submitting and waiting for the CPU-intensive data computation to finish
3. Appending the new testing data to the training or validation data file.
4. Selecting and training the neural network model

These exhaustive tasks make manual data generation and neural network model training very time-consuming, human intensive and error-prone. For this reason, the advanced AMG method is used instead. The process of training and result for each stage is reported in Table 4.4. This example is used to illustrate the advanced AMG technique. Starting from a small network size the neural network structure quickly expands until the accuracy is achieved. AMG then optimizes the structure to obtain the compact network size.

In the first stage, a neural network $h^1(x, w)$ starts from previous training in Chapter 3 with 10 hidden neurons. A total of 1200 and 1500 samples are used. After the training of stage one is completed more weights are added to increase the network size. With the training error obtained from stage 2 and combined with training error obtained from stage 1 the AMS algorithm now can use the Newton's method to predict the next neural network size. By using Equations 3.8 and 3.9 the next prediction weight is calculated to be 266. The next structure (3 11 12 6) is the closest structure having a total weight close to prediction weight. The training is continued to train the next model. During training if the AMG detects overlearning it will automatically drive EM simulator to generate more data. Or if AMG detects the model is stuck at a local minimum it will perturb the current weights of the neural network and continue training. The Newton method is repeated until stage 8 when the training error exceeds the user desired accuracy of 1%. Observing at stage 7 the prediction weight for the next model is 1700 and the training error E_{right} obtained at stage 8 is 0.86%. The training error is below the desire accuracy but the AMG still requests for larger network size. The reason is that in the AMG algorithm we

set the underlearning condition training error equal to 80 percent of the user desired accuracy. So the model accuracy has to exceed 80% of desired accuracy in order to be considered a successful training. At stage 8 when the total weight is increased up to 4268 the training error falls below the 80% of the desired accuracy. When the training error is achieved, AMG will continue to use the binary search as described in Chapter 3 in order to reduce the network size until the optimal size is obtained.

From the results shown in Table 4.4 we observe that in the beginning the training error drops down very quickly but as the accuracy criterion becomes tighter and tighter the rate of improvement slows down. Also, the network size is exponentially increased for a small increased of model accuracy. Another observation from the table 4.4 is that as the network size increases the AMS tends to use more of MLP5 than MLP4. The reason for this is because the MLP5 has more storage capability than MLP4 as the network size grows bigger and bigger. For instance, in this example to get the network size having a total weight of 1741, the MLP3 needs 174 hidden neurons. As compared to MLP4 it only needs 73 hidden neurons. Hence, in this example using 2 hidden neuron layers is more efficient than 1 hidden layer. The training result of the embedded inductor is shown in Figure 4.2.

Table 4.4 The training stage for the spiral inductors using combination of AMG and AMS method

Training stage	W_{left}	W_{right}	E_{left}	E_{right}	Next Prediction Weight	Equipvalent Neural Network Structure
1	0	0	100	100	106	3 10 6
2	0	106	100	0.065791	116	3 11 6
3	106	116	0.065791	0.061806	266	3 11 12 6
4	116	266	0.061806	0.024087	338	3 13 14 6
5	266	338	0.024087	0.018741	484	3 16 18 6
6	338	484	0.018741	0.014106	680	3 20 22 6
7	484	680	0.014106	0.014077	1700	3 35 37 6
8	680	1700	0.014077	0.008632	4268	3 59 61 6
9	1700	4268	0.008632	0.007902	3002	3 49 50 6
10	1700	3002	0.008632	0.007851	2378	3 43 44 6
11	1700	2378	0.008632	0.007806	2048	3 39 41 6
12	1700	2048	0.008632	0.007832	1913	3 38 39 6
13	1700	1913	0.008632	0.007912	1826	3 37 38 6
14	1700	1826	0.008632	0.007827	1784	3 36 38 6
15	1700	1784	0.008632	0.007844	1741	3 36 37 6

Figure 4.2 Comparison of neural network model prediction of S-parameters with original EM *Sonnet-Lite* data for the embedded inductor. Three out of 27 different inductor lengths and 100 frequency samples per length are shown above.

4.4 Development of the Combined Equivalent Circuit and Neural Network Model for Embedded Capacitor Using the Advanced AMG Approach

In section 4.2 and 4.3 we were able to develop an embedded passive neural network model from EM data. However those models cannot be used directly in time-domain circuit simulation and optimization. In this example we will utilize the combined model technique [18][20][59] to develop an embedded passive model to be used in time domain simulation or optimization. The combined model is a hierarchical model of two levels. In the lower level, a neural model maps the geometrical parameters of the passive component into coefficient matrices of equivalent circuit or state equations. In the higher level, we inject the coefficient matrices into the pre-defined equivalent circuit or state equations to compute the EM response in frequency or time domain. These methods combine conventional EM based techniques and the recent neural network approaches to perform simulation and optimization in either frequency or time domain.

This example is for the purpose of illustrating the applicability of using the advanced AMG algorithm to develop a combined model. The advanced AMG is setup to automatically develop a combined model of an embedded capacitor with two inputs length (L : 6 to 20 mils, step size 2 mils), and dielectric permittivity ϵ_{rcap} (ϵ_{rcap} : 10 to 100, step size 10) and frequency (f : 0.1 to 1GHz, step size 0.1GHz; 1 to 20 GHz, step size 1GHz). The output of neural network model is the equivalent circuit component (inductor (L) and capacitor (C) values). The LC are the values used in a user defined equivalent circuit (EC) to produce the 2-port S-parameters to match the EM simulation training data. The EM simulation training data to be generated are the real and imaginary parts of the 2-port S-parameters (i.e. S_{11} and S_{21}). S-parameter training data is obtained from

Sonnet-Lite. Figure 4.3 (a), (b) shows the 3D physical layout and the combined neural network with equivalent circuit of the embedded capacitor respectively.

Figure 4.3 (a) 3D physical layout of an embedded capacitor and (b) The neural network model combined with equivalent circuit.

To develop the combined embedded passive model, ADG is first setup to automatically generate the EM data for the various physical geometries (L and ϵ_{rcap}) of the embedded passive device. Using manual data generation is prohibited because it involves many tasks as described in section 4.3. Once the data is obtained, the LC lumped components are extracted for the pre-defined equivalent circuit that allows for the best map between the S-parameter and the EM data in the frequency bandwidth of interest [18]. Each set of the geometry and lumped components are added to the training and validation data. Once the advanced AMG detects enough data in the training and validation set it will automatically train a neural network model to learn the relationship between the equivalent circuit component values and geometry. If overlearning is detected during neural network model training, AMG will automatically call for data generation and add the new set to either training or validation data. If underlearning is detected, AMS will automatically expand the neural network size by adding more internal weights to the

current network structure. After training, the neural network can accurately calculate the component values of the given equivalent circuit for a given input geometry.

The user specified test error for the neural network model is 1%. The advanced AMG training begins with an initial neural network of 5 hidden neurons. A final neural network model of 12 hidden neurons is able to learn the LC dependence on the input geometry. Combining the equivalent circuit with the neural network the results in an overall model accuracy is 0.86% when compared to the EM simulation results. Figure 4.4 shows a comparison between the real part of S_{11} of the EM simulation results and combined model for a set of geometry inputs.

Figure 4.4 Comparison of RS_{11} of the original EM data (-x-) and the equivalent circuit neural network model (-o-) for two sets of $(L, \epsilon_{\text{rcap}})$ values

4.5 Applying Embedded Components in Frequency Domain Circuit Simulation

In this example, we apply the embedded passives models developed from section 4.2 and 4.3 in the frequency domain circuit optimization for an amplifier circuit design [14]. In order to accomplish this, the embedded passive neural network models are first uploaded into the circuit simulator *Agilent-ADS* [40], using corresponding plug-in software tool at

Carleton University's *NeuroADS* [78]. Once the uploading is finished we can then use these neural network models just as any other non-neural network models available in the simulators library. Figure 4.5 (a) shows the amplifier design using a lump component and Figure 4.5 (b) shows the passive components in the amplifier circuit are replaced by the neural network embedded passive models. The trained embedded passives neural network models can be subsequently used as fast models of resistor and capacitors representing the S-parameter variations with respect to changes in geometrical/physical input parameters. The model is also used to provide EM-level effects of the embedded passive components for circuit simulation and optimization of the amplifier. During circuit optimization every time the geometrical parameters change, neural models are recalled to provide quick estimation of the EM behavior of passive components. This way, repetitive and expensive executions of detailed EM simulation are avoided.

Figure 4.5 The amplifier circuit, in which, all the passive components are represented by the pure neural network models. N_R and N_C denote neural network models of embedded resistors and capacitors respectively.

From the Figure 4.5 (b), N_{Cdc} and N_{Cd} are neural models of the decoupling capacitors of the amplifier. N_{Cc} 's are the neural models of coupling capacitors. N_{R1} , N_{R2} , N_{R3} , and N_{R4} represent neural models of the resistor bias networks, and N_{R5} is the neural model of the resistive load. We constructed a schematic of the circuit in *Agilent-ADS* [40]. The responses of the circuit are obtained in the form of two-port S-parameters by performing small-signal AC analysis.

The parameters considered in optimization are length and width of all the resistors in neural network models. The design specifications of the amplifier outputs are:

Gain	Frequency Range
$2.5 \text{ dB} \leq \text{Gain} \leq 4 \text{ dB}$	for $f \leq 2 \text{ GHz}$
$-3 \text{ dB} \leq \text{Gain} \leq -1 \text{ dB}$	$10 \text{ GHz} < f < 14 \text{ GHz}$

Our objective function is the amplifier gain in the frequency range 1-20 GHz. For every change in the values of the geometrical parameters, the EM effects need to be recalculated. This step is very difficult to achieve in *ADS* without using neural models since the EM responses of the embedded resistor need to be regenerated for different physical parameters of L and W . Using neural network models allows geometrical/physical parameters of the embedded passive to be included as optimization variables. In other words whenever optimization changes R geometry, the corresponding neural network model is called with the new geometrical dimensions as inputs. Hence, it is much easier to incorporate the EM effects into circuit simulation and optimization, which otherwise would be difficult to achieve with direct EM simulators. Table 4.5

demonstrates the changes of the geometrical parameters of resistors with the value after optimization. The circuit responses before and after optimization are shown in Figure 4.6.

Table 4.5 The comparison of geometry of embedded passives before and after optimization in the amplifier circuit.

Design Variable	Before Yield Optimization (mils)	After Yield Optimization (mils)
L_1	9.5	9.89961
W_1	19.4	16.95303
L_2	14.6	15.23131
W_2	8.25	8.6287
L_3	26.7	27.02291
W_3	9.3	10.27157
L_4	17.6	19.70365
W_4	14	15.5325
L_5	30	27.89348
W_5	8.5	10.03643

Figure 4.6 Gain vs. Frequency for *ADS* lumped components (—○—) and the circuit response after optimization (—)

4.6 Applying Embedded Components in Time Domain Circuit Simulation

Signal integrity is an important aspect of high-speed VLSI interconnects design. To demonstrate the use of the passive EM-based neural network models in time domain, we trained two combined equivalent circuit and neural network models of embedded resistor and embedded capacitor using method described in section 4.4. The results of the combined models are subsequently implemented them into *HSPICE* [39] to perform circuit simulation and optimization including geometries of the embedded passive components as design variables [13][14][18][19]. Here we demonstrate an example of signal integrity analysis and optimization of VLSI interconnects in printed circuit board, as shown in Figure 4.7 including nonlinear drivers/receivers, coupled transmission lines and embedded resistors and capacitors whose geometrical values are adjustable.

Figure 4.7 Illustration of VLSI interconnect example for signal integrity analysis in printed circuit board with coupled transmission line, nonlinear buffers, and embedded passives.

A trapezoidal pulse signal is applied to the input buffer of the circuit in Figure 7. The results show that the combined models provide possibility to efficiently adjust the geometry of embedded passives in high-frequency circuit design. Because we used neural models to learn the nonlinear relationship between geometry and coefficient vectors, the geometry becomes variable in circuit design. Table 4.6 shows the models geometrical/physical values before and after optimization.

Table 4.6 Comparison of the Physical/Geometrical Parameters in Embedded Passives Before Optimization and After Optimization.

		Before Optimization (Initial Value)	After Optimization (Design Centre)
Embedded Resistor	Length (mil)	13.50	9.50
	Width (mil)	41.02	62.50
Embedded Capacitor	Length (mil)	38.00	29.50
	ϵ_{rcap}	15.50	13.51

The results in Figure 4.8, there are signal integrity effects such as delay and distortion at output buffer. To improve signal integrity, optimization was performed. The optimization used 136 iterations including repetitive evaluation of combined EC-NN models to reach the criteria of the optimization goal. The total computation time based on our combined EC-NN models was 3.75 minutes. After optimization, the output waveforms have been improved with reduced distortion, delay and crosstalk.

Figure 4.8 Comparison of signals from input buffer and output buffer of the signal integrity example before and after optimization of the embedded passives.

This example demonstrates the capability of the combined EC-NN models to allow the geometries of embedded passives to be design variables in high-speed time domain circuit simulation and optimization, which is a task not yet achieved by the conventional techniques.

Chapter 5

Conclusions and Future Research

5.1 Conclusions

Modeling of embedded passive components is important for electrical and physical design of multilayer printed circuits. The conventional method for developing an embedded passive neural network model, involves several sub-tasks such as data generation, neural network size selection, training, and neural model validation. It is a usual practice to use trial and error to find a suitable number of training data, neural network size, selecting different training algorithm etc,. This is not only time consuming involving lots of human experience, but also may not generate an optimal solution for a given problem. Therefore, this thesis was aimed to develop a complete automatic model generation (AMG) in order to improve neural network based modeling method for an efficient and faster embedded passive model generation.

The advanced AMG technique combines advanced concepts such as the automatic data generation (ADG), automatic multilayer selection (AMS), and automatic neural network training into a unified framework. Starting with zero amounts of training and validation data followed by neural network training in a stage wise manner, each stage can then

involve some or all of the sub-tasks such as error-based adaptive sampling and incremental data generation. To facilitate dynamic data generation, ADG has been utilized to drive the EM data generators such as *Sonnet-lite*. The framework of the AMG also allows efficient use of training data, i.e., the use of more samples in nonlinear-sub regions of the model input space and relatively fewer samples in smooth sub-regions. As for the neural network size adjustment, initially a model can be trained starting with a simple neural network structure. Based on the error criteria at the end of each stage, AMS utilizes the Newton's extrapolation method to adjust the number of layers or the number of neurons in each layer to expand the neural network size until network training converges. Once the network training converges, AMS starts using the binary search algorithm to optimize the network size by reducing the total number of weights. The optimization process is repeated until the network is no longer convergent. During network model training, AMG automatically makes a decision on controlling data samples, selecting training algorithm and model validation at the end of each stage. In a successful automation approach, at the end AMG will produce a compact neural network model that will exceed user desired accuracy using minimum sample data and CPU time. Another advantage of making whole neural network training automatic is to reduce the human time, effort, and user understanding of the neural network issues. Through demonstration of selected examples, the overall AMG algorithms have shown to outperform the conventional step-by-step neural modeling approach in terms of both cost and speed.

5.2 Future Directions

Neural networks are emerging as a powerful technology for RF and microwave characterization, modeling, and design. Neural networks are "generic" in the sense that

they can be potentially achieved at all levels of microwave design from device and components to circuit and systems, and from modeling and simulation to optimization and synthesis. Here, we just use AMG to automatically develop embedded passive component models, but AMG algorithms are not only limited to be used for this purpose. In future, we will be able to use this tool to develop various library models from device to components, and circuit to systems level. The result of library will provide the efficient and accurate models to link commercial microwave EM simulators to be used for circuit design and optimization.

The development of AMG algorithms has addressed the aspect of modeling automation. The next couple of steps are to investigate methodologies to expand the AMG to even higher levels.

1. From the data generation point of view, this thesis has demonstrated the efficient generation of neural models by automatic driving of simulators. In practice, sometimes the simulator results and actual measurements do not agree with each other. The neural network model created using the actual data measurement is more accurate. The robust automatic algorithms driving the computer-controlled measurement equipment to generate the training and validation data can be more practical in future.

2. EM modeling of embedded passive components often requires a large neural network size. In general, network size effects network complexity and learning time, but most importantly, it effects the generalization capability of the network. A good

generalization neural network model has the ability to produce accurate results on data outside its training set. Therefore, a reduction phase was introduced to reduce the network size after the training process. In the current AMS approach, the reduction phase is also applied after network is convergent, but since the network structure created by AMS is a fully connected neural network it may not guarantee the optimum network size. Fully connected neural network has the weights connect every output node to every hidden node. In reality the number of inputs (ie. hidden nodes) that each output requires depends on how complex the decision region is for that output. The higher the complexity of decision region, the higher the number of hidden nodes required by the output. In other words, in multiple-output MLP output nodes are not required to be connected to all hidden nodes. As an example for the embedded resistor was discussed in Chapter 3, the output of the real S_{21} was non linear so it needed more hidden nodes between the input and output. The output of imaginary S_{21} was linear so it required less hidden nodes. Therefore, the training approach that automatically determines when the pruning should start and stop is more desirable.

3. The neural network model created by AMG start with a minimum network size. Neural network must be trained every time after changing its neural network architecture by AMS. The current training approach is to optimize all the weights after the change. The main disadvantage of this approach is that the solution space to be searched becomes too large resulting slower convergence. One approach to solve this problem is to optimize only few weights at a time so that the solution space to be searched is reduced

and computational burden is minimized. To automatically determine when and which nodes input weights are to be frozen during network training is desirable.

Expanding the AMG with those suggestions would make AMG faster and more efficient to use. This tool would enable more microwave engineers to quickly benefit from this technology, and their feedback could further stimulate advanced research in this area.

Bibliography

- [1] D.L. Wilcox, R.F. Huang, and D. Anderson, "The multilayer ceramic integrated circuit (MCIC) technology: Opportunities and challenges," *Int. Symp. Microelectron.*, pp.17-23, 1997.
- [2] A. Fathy Et Al, "Design of embedded passive components in low temperature confined ceramic on metal (LTCC-M) technology," *Int. Microwave Symp.*, pp. 1285-1288, 1998.
- [3] J. Purviance and M. Meehan, "CAD for statistical analysis and design of microwave circuits," *Int. J. Microwave and Millimeter-Wave CAE.*, vol.1, pp. 59-76, 1991.
- [4] A.H. Zaabab, Q.J. Zhang, and M.S. Nakhla, "A neural network models approach to circuit optimization and statistical design," *IEEE Trans. Microwave Theory Tech.*, vol. 43, pp. 1349-1358, 1995.
- [5] F. Filicori, G. Ghione, and C.U. Naldi, "Physics-based electron device modeling and computer-aided MMIC design," *IEEE Trans. Microwave Theory Tech.*, vol. 40, pp. 1333-1352, 1992.
- [6] Q.J. Zhang and K.C. Gupta, *Neural Network for RF and Microwave Design*, Norwood, MA: Artech House, 2000.
- [7] P. Burrascano, S. Fiori, and M. Mongiardo, "A review of Artificial Neural Network applications in microwave Computer-Aided Design", *Int. J. RF and Microwave CAE.*, vol. 9, pp. 158-174, 1999.

- [8] R. Biernacki, J.W. Bandler, J. Song, and Q.J. Zhang, "Efficient quadratic approximation for statistical design", *IEEE Trans. Circuit Syst.*, vol. CAS-36, pp. 1449-1454, 1989.
- [9] P. Meijer, "Fast and smooth highly nonlinear multidimensional table models for device modeling", *IEEE Trans. Circuit Syst.*, vol. 37, pp. 335-346, 1990.
- [10] Q.J. Zhang, F. Wang, and M.S. Nakhla, "Optimization of high-speed VLSI interconnects: A review", *Int. J. of Microwave and Millimeter-Wave CAD.*, vol. 7, pp. 83-107, 1997.
- [11] Q.J. Zhang, M.C.E. Yagoub, X. Ding, D. Goulette, R. Sheffield, and H. Feyzbakhsh, "Fast and accurate modeling of embedded passives in multi-layer printed circuits using neural network approach", *Elect. Components & Tech. Conf.*, pp.700-703, San Diego, CA, 2002
- [12] V.K. Devabhaktuni, M.C.E. Yagoub, and Q.J. Zhang, "A robust algorithm for automatic development of neural network models for microwave applications", *IEEE MTT-S Int. Microwave Symp. Dig.*, pp. 2087-2090, Phoenix, AZ, 2001.
- [13] X. Ding, B. Chattaraj, M.C.E. Yagoub, V.K. Devabhaktuni, Q.J. Zhang, "EM based statistical design of microwave circuits using neural models", *Int. Symp. on Microwave and Optical Technology (ISMOT 2001)*, pp.421-426, Montreal, Canada, June, 2001.
- [14] B. Chattaraj, M.C.E. Yagoub, X. Ding, Q.J. Zhang, "EM based optimization of the microwave circuits by neural models and their application to power distribution and decoupling optimization", *European Microwave Conf. (EuMC 2001)*, pp. 125-128, London, England, Sept., 2001.

- [15] *NeuroModeler Plus v.1.02*, Q.J. Zhang, Dept. of Electronics, Carleton University, Ottawa, Canada.
- [16] F. Wang, and Q.J. Zhang, "Knowledge Based Neural models for microwave design," *IEEE Trans. Microwave Theory Tech.*, vol. 45, pp. 2333-2343, 1997.
- [17] G.L. Creech, B. J. Paul, C.D. Lesniak, T.J. Jenkins, and M.C. Calcaterra, "Artificial Neural Network for fast and accurate EM-CAD of microwave circuits," *IEEE Trans. Microwave Theory Tech.*, vol. 45, pp. 794-802, 1997.
- [18] X. Ding, L. Ton, M.C.E. Yagoub, and Q.J. Zhang, "Time-Domain neural modeling approaches for embedded passives in multilayer printed circuits," *Proc. 20th Annual Review of Progress in Applied Computational Electromagnetics.*, Syracuse, NY, April, 2004.
- [19] L. Ton, Y. Cao, J.J. Xu and Q.J. Zhang "Recent advances in neural based time domain EM modeling and simulation", *10th International Symposium on Antenna Technology and Applied Electromagnetics 2004/URSI.* , Ottawa, Canada, July, 2004.
- [20] L. Ton, J.J. Xu, Q.J. Zhang and R. Sheffield⁺, "Modeling of embedded passives in multi-layer printed circuits using neural networks", *Int. Conf. on Embedded Passives.*, San Jose, California, June 2004.
- [21] P.M. Weston and K.C. Gupta, "EM-ANN models for microstrip vias and interconnects in dataset circuits," *IEEE Trans. Microwave Theory Tech.*, vol. 44, pp. 2495-2503, 1996.

- [22] V.B. Litovski, J. Radjenovic, Z.M. Mrcarica and S.L. Milenkovic, "MOS transistor modeling using neural network," *Electronics Letter.*, vol.28, pp.1766-1768, 1992.
- [23] V.K. Devabhaktuni, C. Xi and Q.J. Zhang, "A neural network approach to the modeling of heterjunction bipolar transistors from S-parameter data", *Euro. Microwave Conf.*, pp.306-311, Amsterdam, Netherlands, 1998.
- [24] K. Shirakawa, M. Shimizu, N. Okubo and Y. Daido, "Structural determination of multilayered large signal neural network HEMT model", *IEEE Trans. Microwave Theory Tech.*, vol. 46, pp. 1367-1375, 1998.
- [25] P. Burrascano, M. Mongiardo, C. Fancelli, and M. Mongiardo, "A neural network model for CAD and optimization of microwave filters", *IEEE. MTT-S Int. Microwave Symp.*, pp. 13-16, Baltimore, MD, 1998.
- [26] Y.H. Fang, M.C.E. Yagoub, F. Wang, and Q.J. Zhang, "A new macromodeling approach for nonlinear microwave circuits based on recurrent neural network", *IEEE Trans. Microwave Theory Tech.*, vol.48, pp. 2335-2344, 2000.
- [27] P. M. Waston and K.C. Gupta, "Design and optimization of CPW circuits using EM-ANN models for CPW components", *IEEE Trans. Microwave Theory Tech.*, vol.45, pp. 2515-2523, 1997.
- [28] C. Christodoulou, A.E. Zooghby, and M. Georgiopoulos, "Neural network processing for adaptive array antennas", *IEEE APS Int. Symp. Dig.*, pp.2584-2587, Orland, FL, 1999.
- [29] A. Veluswami, M.S. Nakhla, and Q.J. Zhang, "The application of neural network to EM-based simulation and optimization of interconnects in high

- speed VLSI circuits”, *IEEE Trans. Microwave Theory Tech.*, vol. 45, pp. 712-723, 1997.
- [30] G. Kothapali, “Artificial neural network as aids in circuit design”, *Microelectronics Journal.*, vol.26, pp.569-678, 1995.
- [31] Q.J. Zhang and M.S. Nakhla, “Signal integrity analysis and optimization of VLSI interconnects using neural network models”, *IEEE Int. Symp. Circuits Systems.*, pp. 459-462, London, England, 1994.
- [32] T. Hong, C. Wang, and N.G. Alexopoulos, “Microstrip circuit design using neural network”, *IEEE Int. Microwave Symp. Dig.*, pp.413-416, Atlanta, Georgia, 1993.
- [33] M. D. Baker, C.D. Himmel, and G.S. May, “In-situ prediction of reactive ion etch endpoint using neural network”, *IEEE Trans. Components, Packaging, and Manufacturing Tech. Part A.*, vol.18, pp.478-483, 1995.
- [34] M. Vai, S. Wu, B. Li, and S. Prasad, “Reverse modeling of microwave circuits with bidirectional neural network models”., *IEEE Trans. Microwave Theory Tech.*, vol.46, pp. 1492-1494, 1998.
- [35] M. Vai and S. Prasad, “Automatic impedance matching with a neural network”, *IEEE Microwave Guided Wave Letter.*, vol. 3, pp. 353-354, 1993.
- [36] *MINIMOS v.6.1*, Institute for Microelectronics, Technical University Vienna, Vienna, Austria.
- [37] *Sonnet-Lite v.9.0*, Sonnet Software Inc., Liverpool, NY, USA.
- [38] *HFSS v.7.0.11*, Ansoft Corporation, Pittsburgh, PA, US.
- [39] *HSPICE v. 2001.2*, Avant! Corporation, Fremont, CA, USA.

- [40] *ADS v 1.5*, Agilent Technologies, Palo Alto, CA, USA.
- [41] *Ensemble*, Ansoft Corporation, Pittsburgh, PA, US.
- [42] V.K. Devabhaktuni, C. Xi, F. Wang, and Q.J. Zhang, "Robust training of microwave neural models", *Int. J. RF and Microwave CAE, Special Issue: Application of ANN's to RF and Microwave Design.*, vol. 12, pp. 109-124, 2002.
- [43] C. Cybenko, "Approximation by superpositions of a sigmoid function", *Math. Control Signals Systems.*, vol. 2, pp.303-314, 1989.
- [44] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximations", *IEEE Trans. Neural Networks.*, vol. 2, pp. 359-366, 1989.
- [45] S. Tamura and M. Tateishi, "Capabilities of a four-layered feedforward neural network: Four layer versus three", *IEEE Trans. Neural Network.*, vol. 8, pp.251-255, 1997.
- [46] D.E. Rumelhart, G.E. Hinton, and R.J. Williams, "Learning internal representation by error propagation", *Parallel Distributed Processing.*, vol. 1, pp.318-362, 1986.
- [47] S. Haykin, *Neural Networks: A Comprehensive Foundation Second Edition*, Prentice Hall, Upper Saddle River, HJ, 1999.
- [48] K. Ochiai, N. Toda, and S. Usui, "Kick-out learning algorithm to reduce the oscillation of weights", *IEEE Trans. Neural Networks.*, vol. 7, pp. 797-807, 1994.

- [49] S.J. Perantonis and D.A. Karras, "An efficient constrained learning algorithm with momentum acceleration", *IEEE Trans. Neural Networks.*, vol. 8, pp. 237-249, 1995.
- [50] *Neural Network Toolbox: For Use with Matlab*, The MathWorks Inc., Natick, Massachusetts, 1993.
- [51] R.A. Jacobs, "Increased rate of convergence through learning rate adaptation", *IEEE Trans. Neural Networks.*, vol. 1, pp. 295-307, 1988.
- [52] S. Haykin, *Neural Networks*, IEEE Press, New York, NY, 1994.
- [53] T. R. Cuthbert Jr., "Quasi-Newton methods and constraints," in *Optimization Using Personal Computers*, New York, NY: John Wiley & Sons, 1987.
- [54] K.R. Nakano, "Partial BFGS update and efficient step-length calculation for three layer neural network", *Neural Computation.*, vol. 9, pp.123-141, 1997.
- [55] S. McLoone and G.W. Irwin, "Fast parallel off-line training and multilayer perceptrons," *IEEE Trans. Neural Networks.*, vol.8, pp.646-653, 1997.
- [56] F. Wang, V.K. Devabhaktuni, C.G. Xi, and Q.J. Zhang, "Neural network structures and training algorithms for RF and microwave application", *Int. J. RF and Microwave Computer-Aided Engineering: Special Issue on Applications of Artificial Neural Network to RF and Microwave Design*, vol. 9, pp.230, 1999.
- [57] M. Stinchcombe and H.White, "Universal approximation using feedforward networks with non-sigmoid hidden layer activation function," *Proc. Conf. Neural Networks.*, pp.613-618, Washington, DC, 1989.

- [58] L. Ton and Q.J. Zhang, "Automatic neural network model generation for EM modeling of embedded passives in multi-layer printed circuits", *Proc. 20th Annual Review of Progress in Applied Computational Electromagnetics.*, Syracuse, NY, April, 2004.
- [59] L. Ton, H. Sharma, J.J. Xu, and Q.J. Zhang, "Automated techniques of neural model development for RF/Microwave CAD", *16th Asia Pacific Microwave Conference* New Delhi, India, December, 2004.
- [60] T.Y. Kwok, D.Y. Yeung, "Constructive algorithms for structure learning in feed forward neural network for regression problem's", *IEEE Transactions on Neural Network.*, vol. 8, no. 3, pp. 630-645, 1997.
- [61] T. Ash, "Dynamic node creation in backpropagation networks", *Connection Science.*, vol. 1, no. 4, pp. 365-375, 1989.
- [62] S. E. Fahlman and C. Lebiere, *The Cascade-Correlation Learning Architecture's*, *Advances in Neural Information Processing System 2*, D. S. Touretzky, Ed. CA: Morgan Kaufmann, pp. 524-532, 1990.
- [63] R. Setiono and L.C.K. Hui, "Use of quasi-Newton method in a feed forward neural network construction algorithm's", *IEEE Trans. Neural Networks.*, vol. 6, pp. 273-277, 1995.
- [64] D.S. Phatak and L. Koren, "Connectivity and Performance Tradeoffs in the Cascade Correlation Learning architecture's", *IEEE Trans. Neural Networks.*, vol. 5, no. 11, pp. 930-935, 1994.
- [65] J. Yang and V. Honavar, "Experiments with the cascade-correlation algorithm," *Microcomputer Applications.*, vol. 17, pp. 40-46, 1998

- [66] M. Lehtokangas, "Modeling with constructive backpropagation", *IEEE Trans. Neural Networks.*, vol. 12, pp. 707-716, 1999
- [67] R. Reed, "Pruning Algorithms – A Survey," *IEEE Trans. Neural Network.*, vol. 4, pp.740-747, 1993.
- [68] J. de Villiers, E. Barnard, "Backpropagation neural nets with one and two hidden layers", *IEEE Trans. Neural Network.*, vol. 4, pp.136-141, 1993.
- [69] R. Martin-Clemente, S. Hornillo-Mellado, I. Acha, "Neural net with two hidden layers for non-linear blind source separation", *Proceedings of the International Joint Conference on Neural Network.*, vol. 1, pp. 726-731, 2003.
- [70] C.A. Perez, A. Holzmann "Optimization of one and two hidden layer neural network architectures for handwritten digit recognition" *IEEE International Conference on Systems, Man and Cybernetics, 1995. 'Intelligent Systems for the 21st Century'*, vol 3, pp.2795 – 2799, 1995.
- [71] X. Zeng, D.S. Yeung, "Sensitivity analysis of multilayer perceptron to input and weight perturbations", *IEEE Trans. Neural Networks.*, vol.12, pp. 1358-1366, 2001
- [72] V.K. Devabhaktuni, M.C.E. Yagoub, Y. Fang, J. Xu, and Q.J. Zhang, "Neural network for microwave modeling: Model development issues and nonlinear modeling techniques", *Int. J. RF Microwave CAE*, vol. 11, pp. 4-21, 2001.
- [73] E. Baum and D. Haussler, "What size net gives valid generalization?" *Neural Computation*, vol.1, pp.151-160, 1989.
- [74] G.B. Huang, "Learning capability and storage capacity of two-hidden-layer feed forward networks", *IEEE Trans. Neural Networks*, vol. 14, pp. 274-281, 2003.

- [75] J. Steward, *Calculus: Early Transcendentals*, Pacific Grove, California: Brooks/Cole, pp. 167-170, 1995.
- [76] J. Hubbard, *Programming With C++*, Schaum's Outlines:McGraw-Hill, pp. 67-69 1996.
- [77] A.C. Reyes, S.M. El-Ghazaly, S.T. Dorn, M. Dydyk, D.K. Schroder, and H. Patterson, "Coplanar waveguides and microwave inductors on silicon substrates", *IEEE Trans. Microwave Theory Tech.*, vol. 43, pp. 2016-2022, 1995.
- [78] *NeuroADS v1.02*, Dr. Q.J. Zhang, Department of Electronics, Carleton University, 1125 Colonel by drive, Ottawa K1S 5B6, Ontario, Ottawa.