A Robust Algorithm for Automatic Development of Neural Network Models for Microwave Applications

Vijay Devabhaktuni, Mustapha C.E. Yagoub, and Qi-Jun Zhang

Department of Electronics, Carleton University, Ottawa, Ontario, K1S 5B6, Canada

***Abstract* — In this paper, we propose a robust algorithm for automating the neural network based RF/Microwave model development process. The algorithm can build a neural model starting with zero amount of training/test data, and then proceeding with neural network training in a stage-wise manner. In each stage, the algorithm utilizes neural network error criteria to determine additional training/test samples required and their location in model input space. The algorithm dynamically generates these new data samples during training, by automatic driving of simulation tools, e.g., *OSA90*, *Ansoft-HFSS*. Initially, fewer hidden neurons are used, and the algorithm adjusts the neural network size whenever it detects under-learning. Our technique integrates all the sub-tasks involved in neural modeling, thereby facilitating a more efficient and automated model building process. It significantly reduces the intensive human effort demanded by the conventional step-by-step neural modeling approach. The algorithm is demonstrated through MESFET and Embedded Capacitor examples.**

I. INTRODUCTION

Recently, a neural network based CAD approach has been introduced for microwave modeling and design [1]. Neural models are developed from microwave data through a process called training. These models are used during microwave design to provide fast estimation of device/circuit behaviors [2][3]. Neural modeling techniques have been applied to a wide variety of microwave problems, e.g., transistors [2], transmission lines [3], vias [4], CPW components [5], filters [6], amplifiers [7]. Significant speed-up of CAD by using neural models in place of CPU-intensive EM/physics models resulted in a drive to develop advanced neural modeling techniques.

Neural model development involves data generation, preprocessing, training, and testing. Conventionally, these sub-tasks are carried out separately in a sequential manner. Such an approach could demand intensive human effort, e.g., exhaustive data generation using commercial simulators, repetitive training's with different neural network sizes until desired model accuracy is achieved etc. Since human decisions are involved in the conventional approach, a reasonable understanding of issues like "How much data?", "How many neurons?", is also necessary. As such, automation of neural modeling process could be of immense interest, because it transfers human workload to the CPU.

In this paper, we propose a novel algorithm that automatically drives all the sub-tasks involved in neural modeling process in a unified way. Entire model development process right from data generation to neural model testing is integrated and computerized. The algorithm facilitates periodic communication between various sub-tasks, thus enabling adjustment or enhancement in the execution of a sub-task based on the feedback from other sub-tasks. Neural model development process can start with zero amount of training/test data samples and with a small neural network. As the stage-by-stage training continues, the algorithm can (i) Determine the number of additional training/test samples required and their distribution in model input parameter space based on neural network test error, (ii) Adjust neural network size (i.e., add more hidden layer neurons) based on neural network training error.

The algorithm has built-in simulation drivers (e.g., *OSA* driver, *Ansoft-HFSS* driver) for automatic driving of 3D-EM or physics-based simulation tools during neural network training. From the user's perspective, the technique establishes a quantitative link between neural model accuracy, the number and distribution of training/test data, and the neural network size. The algorithm identifies nonlinear sub-regions in the model input space (if any) and adds relatively more samples in such regions, while fewer samples are generated in smooth regions, thus making judicious use of data.

II. PROPOSED AUTOMATION ALGORITHM

A. Methodology

Let $\boldsymbol{x}$ and $\boldsymbol{y}$ represent n- and m-dimensional input and output vectors of a microwave problem respectively, and $\boldsymbol{y}(\boldsymbol{x})$ and $\tilde{\boldsymbol{y}}(\boldsymbol{x},\boldsymbol{w})$ represent physics/EM and neural network relationships between $\boldsymbol{x}$ and $\boldsymbol{y}$, where $\boldsymbol{w}$ denotes the neural network weight vector. Let L_k and T_k represent training and test data sets during k^{th} training stage. To

begin with, the algorithm considers the original bounded n-dimensional input space of interest (R_0) as a group of 2^n regions (hyper-cubes) R_1, R_2,…, R_{2^n}, of equal-volume. Initial training and test data are systematically generated for each region in a pre-defined way (e.g., central composite distribution), and an initial neural network with relatively fewer hidden layer neurons is used. After k^{th} training stage, if the neural model accuracy does not match the user-desired accuracy, the worst test sample with maximum error is identified by,

$$\boldsymbol{x}_k^* = \arg \max_{\boldsymbol{x} \in T_k} \left\| \tilde{\boldsymbol{y}}(\boldsymbol{x}, \boldsymbol{w}_k) - \boldsymbol{y}(\boldsymbol{x}) \right\| . \quad (1)$$

The worst region R_k^* to which $\boldsymbol{x}_k^*$ belongs is further divided (split) into 2^n new regions. L_k and T_k are updated by generating incremental training data (p new samples) and test data (q new samples) in these new regions as,

$$L_{k+1} = L_k \bigcup \left\{ \boldsymbol{x}_k^* + \boldsymbol{P}_i \frac{\boldsymbol{x}_{\max} - \boldsymbol{x}_{\min}}{2^{r+1}}, i = 1,2,..., p \right\}, \quad (2)$$

$$T_{k+1} = T_k \bigcup \left\{ \boldsymbol{x}_k^* + \boldsymbol{Q}_j \frac{\boldsymbol{x}_{\max} - \boldsymbol{x}_{\min}}{2^{r+2}}, j = 1,2,..., q \right\}, \quad (3)$$

where, $\boldsymbol{x}_{\max}$ and $\boldsymbol{x}_{\min}$ are the extreme boundaries of R_0, r is the number of splits after which R_k^* was formed, and $\boldsymbol{P}_i$ and $\boldsymbol{Q}_j$ are n x n diagonal matrices. Each diagonal element of these matrices could take one of the values 0, +1, or –1, depending upon the pre-defined sample distribution in each hypercube. In the next stage, the algorithm trains the neural network with L_{k+1} and tests the neural model using T_{k+1}. The algorithm also monitors training error and its gradient to detect possible situations of under-learning of the neural network. As a remedy to under-learning, additional hidden layer neurons are added to the neural network. The process is continued until user-desired model accuracy is achieved. A flow-chart of the proposed automatic neural modeling algorithm illustrating the k^{th} stage of neural model development is shown in Fig. 1.

B. Implementation

Let $\Re$ represent a set of regions and E_d represent user-desired neural model accuracy (average test error). Let $E_l(k)$, $E_t(k)$, $\Delta E_l(k)$ represent training error, test error, and gradient of the training error, for the neural network structure S_k at the end of k^{th} stage of training, and k_{max} represent the maximum number of stages set by the user. N_k denotes the number of hidden layer neurons in S_k. In addition, there are user-inputs k_c, k_u, ΔE, and α. The pseudocode of the proposed algorithm is presented below.

Initialization: $k = 1$, $N_1 = N_0$, $L_1 = \{ \boldsymbol{x}_i \mid \boldsymbol{x}_i$ is a vertex of $R_0\}$ and $T_1 = \{ \boldsymbol{x}_j \mid \boldsymbol{x}_j$ is the center of $R_0\}$. Generate $\boldsymbol{y}_i$ and $\boldsymbol{y}_j$ for all $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$. There is only one region, $R_0 \in \Re$. Train initial neural network S_1 using the samples $(\boldsymbol{x}_i, \boldsymbol{y}_i)$, $\boldsymbol{x}_i \in L_1$, and test the neural model with $(\boldsymbol{x}_j, \boldsymbol{y}_j)$, $\boldsymbol{x}_j \in T_1$. *Activate Data Generation* (R_0).

Activate Data Generation (R): Split the region R into 2^n new regions of equal volume. Delete the region R from $\Re$ and add the new regions to $\Re$. Generate training and test data for new regions by automatically driving the simulator. Update L_k and T_k to include the new samples and go to *Automatic Training*.

Automatic Training: $k = k+1$. Train neural network structure S_k using data samples in L_k. Test the neural model with samples in L_k to obtain training error $E_l(k)$, and with samples in T_k to obtain test error $E_t(k)$.

if ($k > k_{max}$) or (($E_t(k) \le E_d$) for k_c consecutive training stages), ***Stop Training***

else if ($E_t(k) > \alpha E_d$) and (($\Delta E_l(k) < \Delta E$) for k_u consecutive training stages) then under-learning is detected. ***Add Hidden Neurons****, i.e.*, $N_k = N_k + \delta$, where δ is number of newly added hidden neurons in S_k.

else ***Choose Worst-Performing Region*** R_k^* following eq. (1). *Activate Data Generation* (R_k^*).

III. EXAMPLES

The proposed algorithm is implemented and incorporated into our *NeuroModeler* software [8].

A. Automatic Generation of MESFET Neural Model

In this example, neural model of a MESFET is developed. The input space $\boldsymbol{x}$ contains gate-length (l), channel thickness (a), gate-source voltage (v_g), and drain-source voltage (v_d). Drain current (i_d) is the neural network output y. The complexity of the input-output relationship is not known a priori and so is the number of hidden layer neurons required. For a given E_d, the number of training/test samples needed and their distribution in (l, a, v_g, v_d) space are also not known.

The automatic algorithm starts with an initial model structure (3-layer MLP with 9 hidden neurons) and with zero training and test data. In the first stage, the algorithm generates 16 samples to train the neural network and 1 sample to test it. During subsequent training stages, incremental training/test data and additional hidden neurons are automatically added as needed. The online data generation during the model building process is achieved by our Osa90 driver, which automatically drives the *OSA90* simulator [9]. For E_d = 1%, the algorithm produces a neural model with 0.82% accuracy after 5 training stages. A total of 275 training samples and 61 test samples are used and the final neural model has 16 hidden neurons. Using the manual step-by-step neural modeling

approach, a MLP neural network with 16 hidden neurons, trained using 275 uniform-grid samples yielded a neural model with 1.5% test error. The neural model obtained by our algorithm is further tested with a large set of test data (80000 samples) never seen during training. The average test error is observed to be 0.90%, thus verifying the reliability of our algorithm. The automatic data generation algorithm tends to give more accurate neural models with same amount of training data, as compared to the conventional uniform-grid approach as shown in Table I. This is because the proposed method uses efficient distribution, i.e., more (less) data are generated in nonlinear (smooth) regions.

B. Automatic Generation of Embedded Square Capacitor Neural Model

Accurate modeling of 3-D EM behaviors of embedded components used in high-speed high-frequency multi-layer printed circuit boards (PCB) is necessary for efficient CAD. In this example, neural model of an embedded square capacitor shown in Fig. 2 is developed. The input space $\boldsymbol{x}$ contains length/width (l), thickness (t), dielectric constant of the capacitor (ε_r), and frequency (f), $f \in$ (0.1 – 20GHz). Real and imaginary parts of S_{11} are the model outputs $\boldsymbol{y}$. In the first stage, the neural network S_1 has 12 hidden neurons, and 16 training samples and 1 test sample are used. Whenever, the automatic algorithm needs more data, it dynamically drives the *ANSOFT-HFSS* simulator [10] using our Ansoft-Hfss driver. After 10 stages of training, the final neural model S_{10} with 20 hidden neurons, 554 training, and 136 test samples achieved an accuracy of 0.92%. On the other hand, a MLP neural network with 20 hidden neurons trained with 554 uniform grid samples has a test error of 6.8%. A comparison of training data shows that the automatic algorithm uses 176 training samples in the sub-region $f \in$ (0.1 – 3GHz) where data is relatively difficult to learn, while the manual uniform-grid sampling uses only 128 data points. Using uniform-grid and manual training approach, 768 training samples are required to achieve a model accuracy of 1%. Neural model generated by our algorithm is subjected to an independent test with a large set of data (8200 samples) never seen during training and the test error is observed to be 1.04%, confirming the reliability of our model.

A time comparison between the proposed algorithm and the step-by-step manual neural modeling approach is shown in Table II. It can be seen that the human time required in the case of the proposed algorithm is very small as compared to the manual approach. The reason is that, in the manual approach, whenever an input parameter (e.g., length) changes, one must manually re-draw capacitor, run *HFSS*, and update the data files. In our algorithm, data generation is automatic. The CPU time required by our approach is also relatively smaller. This is because the manual approach requires more data and neural networks of different sizes are to be trained before a neural model with desired accuracy is achieved.

IV. Conclusion

We proposed a robust algorithm for automatic development of neural network based RF/Microwave models. The algorithm can build neural models starting with small amounts of training/test data and then proceeding with stage-by-stage training. The algorithm generates new data samples during training, by automatic driving of simulation tools. Neural network size can also be adjusted during training. The technique provides a quantitative link between the neural model accuracy, the number and distribution of training data, and the neural network size. It can be seen from the examples that the proposed algorithm uses relatively fewer samples than the manual approach to achieve similar model accuracy. A significant reduction in the human time and effort is demonstrated in the capacitor example.

References

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

[2] F. Wang and Q.J. Zhang, "Knowledge-based neural models for microwave design", *IEEE Trans. Microwave Theory Tech.*, vol. 45, pp. 2333-2343, 1997.

[3] F. Wang, V.K. Devabhaktuni, and Q.J. Zhang, "A hierarchical neural network approach to the development of a library of neural models for microwave design", *IEEE Trans. Microwave Theory Tech.*, vol. 46, pp. 2391-2403, 1998.

[4] P.M. Watson, K.C. Gupta, and R.L. Mahajan, "Applications of knowledge-based artificial neural network modeling to microwave components", *Int J. RF and Microwave CAE*, vol. 9, pp. 254-260, 1999.

[5] P. Watson, G. Creech, and K. Gupta, "Knowledge based EM-ANN models for the design of wide bandwidth CPW patch/slot antennas," *1999 IEEE APS-S Int. Symp. Dig.*, Orlando, FL, 1999, pp. 2588-2591.

[6] J.W. Bandler, M.A. Ismail, J.E. Rayas-Sanchez, and Q.J. Zhang, "Neuromodeling of microwave circuits exploiting space-mapping technology", *IEEE Trans. Microwave Theory Tech.*, vol. 47, pp. 2417-2427, 1999.

[7] M. Vai and S. Prasad, "Neural networks in microwave circuit design - Beyond black box models," *Int J. RF and Microwave CAE*, vol. 9, pp. 187-197, 1999.

[8] *NeuroModeler v.* 1.3, Q.J. Zhang, Dept of Electronics, Carleton University, Ottawa, Canada.

[9] *OSA90*, Optimization Systems Associates, Dundas, Canada, Now HP EEsof, Santa Rosa, CA, USA.

[10] *Ansoft HFSS v.*7.0.11, Ansoft Corporation, Pittsburgh, PA, USA.

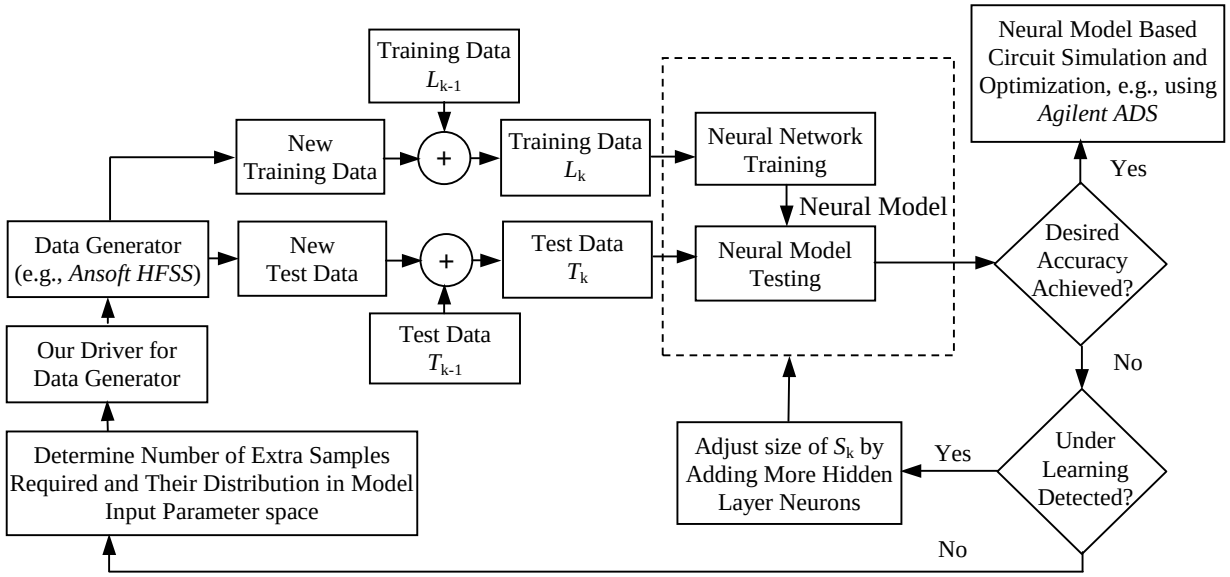

Fig. 1. Flow-chart of the proposed automatic neural modeling algorithm showing the k^{th} stage of model development.

TABLE I. MODEL ACCURACY COMPARISON BETWEEN THE PROPOSED AUTOMATIC MODEL BUILDING ALGORITHM AND THE MANUAL APPROACH BASED ON CONVENTIONAL GRID DISTRIBUTION FOR THE MESFET EXAMPLE. MODEL ACCURACY REPORTED IN THE TABLE IS BASED ON THE TEST RESULTS USING AN INDEPENDENT SET OF 80000 SAMPLES.

Proposed Automatic Algorithm			**Manual Neural Modeling Approach**	
Stage No. (k)	No. of Training Samples	Model Accuracy	No. of Training Samples Used	Model Accuracy
1	16	23.38%	16	25.74%
2	81	6.64%	81	8.05%
3	146	4.32%	144	6.42%
4	211	1.82%	225	3.20%
5	256	0.90%	256	1.28%
6	318	0.48%	320	0.75%

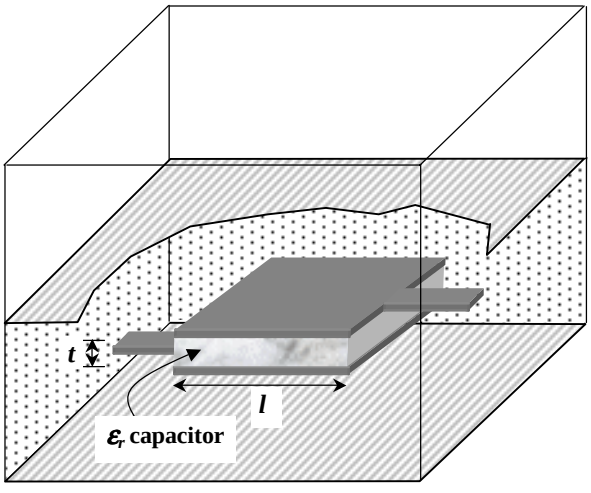

Fig. 2. Embedded capacitor used in multi-layer PCB's. S-parameter neural model of the capacitor is developed from 3D-EM data of *ANSOFT-HFSS* using the proposed algorithm.

TABLE II. COMPARISON OF THE TIME TAKEN BY PROPOSED ALGORITHM AND MANUAL STEP-BY-STEP NEURAL MODELING FOR THE EMBEDDED CAPACITOR.

	Proposed Automatic Model Development Algorithm		**Manual Step-By-Step Neural Modeling Approach**	
	Human Time	CPU Time	Human Time	CPU Time
	5 min	1158 min	498 min	1625 min
Total Time	1163 min		2123 min	