Journal of Engineering Science and Technology
Vol. 13, No. 3 (2018) 625 - 639

EYE-HEIGHT/WIDTH PREDICTION USING ARTIFICIAL NEURAL NETWORKS FROM S-PARAMETERS WITH VECTOR FITTING

CHAN HONG GOAY, PATRICK GOH*,
NUR SYAZREEN AHMAD, MOHD FADZIL AIN

School of Electrical and Electronic Engineering, Engineering Campus,
Universiti Sains Malaysia, 14300 Nibong Tebal, Pulau Pinang, Malaysia
*Corresponding Author: eepatrick@usm.my

Abstract

Artificial neural networks (ANNs) have been used to model microwave and RF devices over the years. Conventionally, S-parameters of microwave/RF designs are used as the inputs of neural network models to predict the electrical properties of the designs. However, using the S-parameters directly as inputs into the ANN results in a large number of inputs which slows down the training and configuration process. In this paper, a new method is proposed to first disassemble the S-parameters into poles and residues using vector fitting, and then the poles and residues are used as the input data during configuration and training of the neural networks. Test cases show that the ANN trained using the proposed method is able to predict the eye-heights and eye-widths of typical interconnect structures with minimal error, while showing significant speed improvement over the conventional method.

Keywords: Artificial neural network, S-parameters, Eye-height, Eye-width, Vector fitting.

1. Introduction

The analysis of signal quality in interconnects is an integral part of the design of any modern electronic device. Faster signalling speed, compounded with smaller design densities, have led to significant signal deterioration due to previously negligible effects such as crosstalk, dispersion, attenuation, reflection and delay [1]. These signal integrity effects, are often analyzed in the time domain, where merits on the signal quality are evaluated in terms of the bit error rate (BER) and the quality of the overall eye diagram, which is normally measured in terms of the width and height of the eye opening. However, simulations of eye diagrams and the subsequent evaluation of the BER is a very resource consuming process as the

Nomenclatures

A	Attenuation of a lossy transmission line, dB
a_n	Poles in a rational function
c_n	Residues in a rational function
d	Constant term in a rational function
E	Electrical length of a transmission line, deg.
E_h	Eye-height
E_w	Eye-width
$e_{n,ij}$	Normalized error of the *j*th element at the *i*th output neuron
$E_{n,Tr}$	The total normalized training error
h	Proportional term in a rational function
L	Physical length of a lossy transmission line
N	Order of approximation for vector fitting
N_O	Number of output neurons
N_{Tr}	Length of training data
R_n^2	Overall normalized performance of a neural network
R_{n,E_v}^2	Normalized evaluation performance of a neural network
$R_{n,Tr}^2$	Normalized training performance of a neural network
S_{xy}	S-parameter measured from port-*x* to port-*y*
$t_{n,ij}$	Normalized target of the *j*th element at the *i*th output neuron
$\mu_{n,i}$	Mean value of $t_{n,ij}$
$y_{n,ij}$	Normalized output of the *j*th element at the *i*th output neuron
Z	Characteristic impedance of a transmission line, ohm

Abbreviations

ADS	Advanced Design System
ANN	Artificial Neural Networks
BER	Bit Error Rate
MLP	Multilayer Perceptron
PRBS	Pseudo-Random Bit Sequence

eye diagram is formed by the overlapping of multiple transient simulations with different transitions and various bit profiles, a long simulation of a pseudo-random bit sequence is normally required which could contain millions of bits. This is further compounded by the fact that a typical design process requires multiple simulations to cover all the design corners and test cases. As a result, there is a constant need for faster and more efficient methods to evaluate the eye diagrams of various interconnect structures.

Recently, modeling process based on the artificial neural networks (ANN) has received considerable attention [2]. ANN is attractive due to its ability to predict highly nonlinear effects and its computational efficiency in providing results once properly trained. This allows the network to be reused over and over again with minimal costs, thus saving valuable time in the design process. However, most efforts on using ANN for interconnects (and its underlying components in RF and microwave) modeling and simulations starts from the design parameters such as the circuit component values and design dimensions [3-6]. This places a constraint on the method as the inputs would be limited to the chosen components and dimensions. In addition, it is also common for more complex designs to

incorporate structures whose behavior are described by a broadband input-output response, normally in the form of S-parameters, instead of the actual dimensions, as this protects any propriety information held by the device. As a result, ANN based modeling from S-parameters has also been recently proposed [7]. The main challenge in using the terminal responses such as the S-parameter as the inputs to the ANN lies in the fact that these responses are frequency dependent. In order to cover the design spectrum, the S-parameters must be considered across a wide frequency range. This results in an exorbitant number of inputs to the ANN which consequently slows the network's training and evaluation process, thus limiting its usability.

In this paper, we propose a new framework for the application of ANN for eye-height and eye-width prediction from S-parameters by using vector fitting. Vector fitting is a popular method in the RF and microwave community which is normally used to extract the dominant poles and residues of a function, which is then used in further modelling processes. In fact, vector fitting has been used together with ANN in microwave modelling problems as reported in [8] where neural models were used in patch antenna modelling. Instead of predicting the S-parameters directly, neural models were developed to predict the poles and residues. The predicted poles and residues are then converted into S-parameters using a rational transfer function. More recent advancements in this research area can be found in [9, 10].

In this work, we first apply vector fitting to decompose the S-parameters into its poles and residues. These poles and residues are then used as the inputs to the ANN. This has the benefit of reducing the number of inputs significantly and also avoids the frequency dependency of the input data (inherent in S-parameters), which results in faster and more efficient simulations. We note that this is different to the approach taken in [8].

The remainder of this paper is organized as follows. Section 2 details the background information on ANN and vector fitting and the proposed methodology. Section 3 shows two numerical examples to validate the proposed method and the results are compared to the conventional method. Finally, a conclusion and future works are presented in section 4.

2. ANN and Vector Fitting

2.1. ANN training algorithm

Artificial neural networks (ANN), especially those based on multilayer perceptron (MLP) have been used in RF and microwave circuits design and modelling as an alternative to conventional methods [11]. As the design of RF and microwave devices becomes more complex, the simulations during the design process will use up a significant amount of time and computational power. Neural network methods are attractive alternative methods because a trained neural network can provide accurate and fast answers to the problem. A common structure for an MLP with an input layer, a single hidden layer and an output layer is shown in Fig. 1.

Before a neural network can be used, it must be trained. During the training process, the weights and biases of the neural network are adjusted to produce outputs as close as possible to the training data targets. This is done by calculating

training errors, which are the differences between the outputs and the targets, which are then used to update the weights and biases during every iteration/cycle of the training process. In this work, the Levenberg-Marquardt backpropagation technique is used as the training function. Other training algorithms are available and the reader is referred to [12] for a comparison of widely used methods.

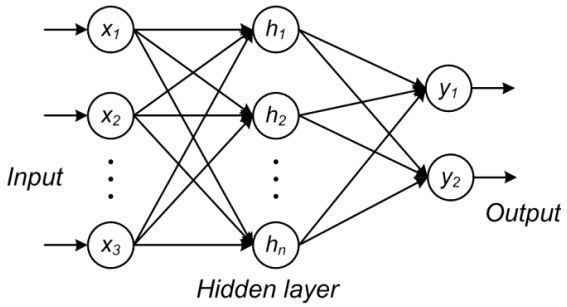


Fig. 1. MLP with an input layer, one hidden layer and an output layer.

In this work, during training, validation and testing, a mean squared normalized error performance function is used to evaluate the network. The total normalized training error $E_{n,Tr}$ is evaluated as

$$E_{n,T_r} = \frac{1}{N_O \times N_{T_r}} \sum_{i=1}^{N_O} \sum_{j=1}^{N_{T_r}} \left(\frac{y_{n,ij} - t_{n,ij}}{2} \right)^2 \tag{1}$$

where N_O is the number of output neurons, N_{Tr} is the length of training data, and $y_{n,ij}$ and $t_{n,ij}$ are the normalized outputs and targets of the jth element in the training data at the ith output neurons, which will be discussed in the next section. The aim of the training process is to minimize $E_{n,Tr}$ by adjusting the weights and biases.

2.2. Normalization of data

The outputs of the neural network are not normalized. One of the main reasons is because it is more convenient to have a neural network model that predicts the eye-height and eye-width directly as its outputs without a de-normalization process at the end. However, the network's errors are normalized between -1 and +1 during the weight and biases adjustment of the training process as shown below:

$$e_{n,ij} = \frac{y_{n,ij} - t_{n,ij}}{2} \tag{2}$$

$$y_{n,ij} = \frac{y_{ij} - \frac{y_{max,i} + y_{min,i}}{2}}{\frac{y_{max,i} - y_{min,i}}{2}} \tag{3}$$

$$t_{n,ij} = \frac{t_{ij} - \frac{t_{max,i} + t_{min,i}}{2}}{\frac{t_{max,i} - t_{min,i}}{2}} \tag{4}$$

where $e_{n,ij}$ is the normalized error, $y_{max,i}$ and $y_{min,i}$ are the maximum and minimum values of the estimated outputs y_i, and $t_{max,i}$ and $t_{min,i}$ are the maximum and minimum values of the target t_i at the ith output neuron.

The normalization process is done because the values of the eye-heights and eye-width fall between different ranges and using the un-normalized data will result in errors of one parameter dominating the performance of the network over the other. Hence, normalizing the eye-height errors and the eye-width errors between -1 and +1 ensures that both errors play equally important roles in the layer's weights and biases adjustment. The evaluation errors, evaluation outputs, and evaluation targets are also normalized in the same approach as in Eqs. (2), (3), and (4).

2.3. Model's performance evaluation

In order to evaluate the performance of a neural network model, training data and evaluation data are both used. The network's performance is calculated by using the coefficient of determination R^2. The normalized training performance R^2_{n,T_r} and normalized evaluation performance R^2_{n,E_v} are given as follows:

$$R^2_{n,T_r} = 1 - \frac{\sum_{i=1}^{N_O}\sum_{j=1}^{N_{T_r}}(y_{n,ij} - t_{n,ij})^2}{\sum_{i=1}^{N_O}\sum_{j=1}^{N_{T_r}}|t_{n,ij} - \mu_{n,i}|^2} \tag{5}$$

$$\mu_{n,i} = \frac{1}{N}\sum_{j=1}^{N_{T_r}} t_{n,ij} \tag{6}$$

$$R^2_{n,E_v} = 1 - \frac{\sum_{i=1}^{N_O}\sum_{j=1}^{N_{E_v}}(y_{n,E_v,ij} - t_{n,E_v,ij})^2}{\sum_{i=1}^{N_O}\sum_{j=1}^{N_{E_v}}|t_{n,E_v,ij} - \mu_{n,E_v,i}|^2} \tag{7}$$

$$\mu_{n,E_v,i} = \frac{1}{Ne}\sum_{j=1}^{N_{E_v}} t_{n,E_v,ij} \tag{8}$$

where $y_{n,ij}$ and $y_{n,Ev,ij}$ are the normalized training output and normalized evaluation output of the ith output neuron and jth element of all the training/evaluation outputs, $t_{n,ij}$ and $t_{n,Ev,ij}$ are the normalized training target and normalized evaluation target of the ith output neuron and jth element of all the training/evaluation targets, and $\mu_{n,i}$ and $\mu_{n,Ev,i}$ are the mean values of $t_{n,ij}$ and $t_{n,Ev,ij}$ respectively.

A good network must be able to produce good results for both the training data and the evaluation data. The aim of the performance evaluation process is to avoid over-learning and under-learning by filtering out models with bad performances. The overall normalized performance of a neural network model R^2_n is given as:

$$R^2_n = \frac{W_{T_r} \times R^2_{n,T_r} + W_{E_v} \times R^2_{n,E_v}}{W_{T_r} + W_{E_v}} \tag{9}$$

where W_{T_r} and W_{E_v} are positive constants that control the weighting of the performance assessment based on the training and evaluation data.

2.4. Vector fitting for input size reduction

While it is beneficial to be able to train and use the ANN based on the terminal responses, using the S-parameters directly as inputs will result in a network with a large number of inputs which will significantly slow down the training and evaluation process. In this work, we apply the use of vector fitting to decompose the input S-parameter into its poles and residues before they are fed into the ANN.

Vector fitting is a popular model order reduction technique where any rational function, *f(s)*, where $s = j\omega$, the complex frequency, can be modelled as

$$f(s) = \sum_{n=1}^{N} \frac{c_n}{s - a_n} + d + sh \tag{10}$$

where N is the order of approximation, c_n are the residues, a_n are the poles and d and h are the constant and proportional terms respectively. In cases where the function does not have an asymptotic value, d and h can be set to zero which reduces the function to

$$f(s) = \sum_{n=1}^{N} \frac{c_n}{s - a_n} \tag{11}$$

Then, through a solution of an overdetermined set of equations, the unknown poles and residues are solved for in an iterative fashion. The reader is referred to [13] for the full formulation, and subsequent improvements in [14-16]. We note that the order, N will control the accuracy of the approximation where higher orders will lead to better accuracy in fitting. However, increasing the order of approximation will also lead to an increase in the number of residues and poles, which ultimately increase the number of input neurons of the neural network model. Hence, the order of approximation should be small while still maintaining a high fitting accuracy.

2.5. Summary of overall proposed framework

A summary of the overall framework for using the proposed method for the modelling and analysis of interconnects is presented in Algorithm 1. We note that all of the steps are described in detail in the previous subsections.

Algorithm 1: Framework of proposed method

1. Determine the input and output parameters
   - Define the distributions of the parameters.
   - Generate the training data (S-parameters) based on the desired parameters and distributions.
2. Input space reduction using vector fitting
   - Solve for the poles and residues in Eq. (11) from the S-parameters by using the vector fitting process.
3. Train the ANN

- Train the ANN (Levenberg-Marquardt backpropagation) using the poles and residues.
- Normalize the errors using Eqs. (2)-(4).
- Update weight and biases based on the normalized error in Eq. (1).

4. Evaluate the neural-network model
   - Generate evaluation data.
   - Verify the performance of the neural network using Eq. (9).
5. Use the ANN to generate the desired outputs.

3. Numerical Results

In this section, two numerical cases which are a lossless transmission line and a lossy transmission line will be presented to illustrate the proposed method. The transmission lines represent the most basic structures of high frequency interconnects. Keysight's Advanced Design System (ADS) [17], will be used to generate the test data and to perform full eye diagram simulations for benchmarking.

The proposed method is implemented according to Algorithm 1. Test data is generated by varying the physical/geometrical parameters of the transmission lines and simulating the corresponding eye-height and eye-width. Then, the data is separated into training data and testing data. After that, the architecture of the neural network and its training algorithm/function are set. Next, the neural network is trained using the training data. Finally, its performance is evaluated using the evaluation data. All the test functions are implemented in MATLAB running on an Intel® Xeon® CPU E5-1620 v3 @ 3.50 GHz with 8GB of RAM.

3.1. Case 1: Lossless transmission line

In this example, a lossless transmission line is simulated in the setup shown in Fig. 2. During simulation, the characteristic impedance, Z and electrical length, E are varied to generate the test data. Z value varies from 10 Ω to 220 Ω with a uniform step of 10 Ω while E is varied from 30° to 180° (at 1 GHz) with a uniform step of 15°. The input signal is a pseudo-random bit sequence (PRBS) at 1 GHz with a minimum voltage of 0 V and maximum voltage of 1 V. In each case, the eye diagram is simulated in ADS and the corresponding eye-height and eye-width is measured at the output. An example eye diagram from the system is shown in Fig. 3.

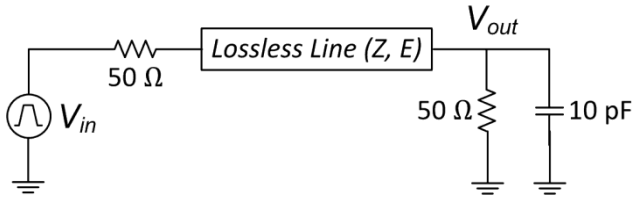


Fig. 2. Circuit in Case 1.

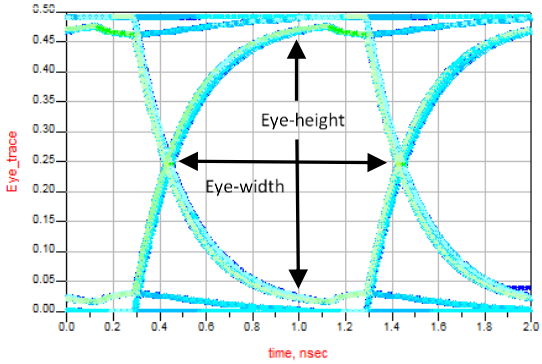


Fig. 3. Example eye diagram showing the eye-height and eye-width.

Then, ANN networks are used to model the system. For the purpose of comparison, the S-parameters of the transmission line are first used directly as the input to the ANN. This represents the conventional method without the use of vector fitting. The S-parameters are obtained from 0 GHz to 5 GHz with a step of 0.01 GHz with the reference impedance set to 50 Ω. Since the modelled system is symmetric, S_{11} is identical to S_{22}. In addition, since the system is also reciprocal, S_{12} is identical to S_{21}. This allows S_{22} and S_{21} to be removed from the inputs thus reducing the total number of inputs by half. An example plot of the S-parameters is shown in Fig. 4.

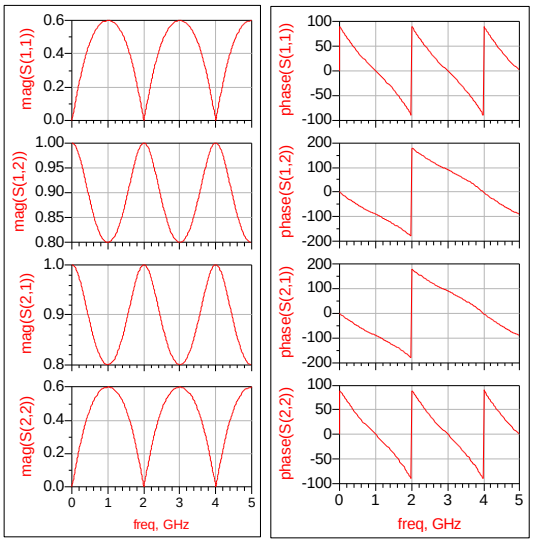


Fig. 4. Example S-parameter of a lossless line.

Nguyen-Widrow initialization algorithm is used to initialize the layer's weights and biases [18]. This algorithm chooses values in order to distribute the active region of each neuron in the layer approximately evenly across the layer's input space. In this work, among all the collected data, 70 to 85 percent of the data is used for training, and the rest of data is used for performance evaluation. The training data is further divided into 3 subsets at random, which are training

set (70%), validation set (15%) and testing set (15%) during the training process. The aim of this separation is to further improve the generalization capability of the neural network and to avoid over-fitting.

The training set is used to compute the gradient and update the network's weights and biases. The validation set is used to end the training process when the validation error fails to decrease continuously for 6 iterations. This method is known as early stopping. The testing set is used for an unbiased performance assessment of the neural network.

In order to investigate the performance of the ANN in this case, ten different networks, each with a number of hidden neurons from 1 to 10 is trained to model the system. In addition, each model is trained 10 times, each time with newly initialized weights, and the average training time and performance is recorded as shown in Table 1. We see that the networks trained from the S-parameters directly require long training times and show large inaccuracies. For example, the case of 10 hidden neurons requires 6418 seconds to train each model. To test 100 different models would have required over 7 days of computing time. This illustrates the problem of using the S-parameters directly as the input to the ANN due to the large number of inputs (1002 complex valued numbers in this case).

Table 1. Details of all neural networks for Case 1 (conventional method).

No. of Hidden Neurons	No. of Trained Models	Total Training Time (s)	Training Time per Model (s)	Average Training Perfor-mance	Average Evaluation Perfor-mance	Best Overall Perfor-mance
1	10	51.59	5.16	0.7761	0.2985	0.8434
2	10	311.52	31.15	0.7778	0.4197	0.8327
3	10	773.08	77.31	0.8671	0.5116	0.7599
4	10	1998.60	199.86	0.8991	0.0866	0.8248
5	10	3989.10	398.91	0.9294	0.1218	0.8149
6	10	4556.20	455.62	0.9276	0.1452	0.7053
7	10	6612.80	661.28	0.8982	0.2223	0.7084
8	10	10695.00	1069.50	0.9107	-0.0946	0.6780
9	10	62616.00	6261.60	0.9150	-0.0043	0.7101
10	10	64187.00	6418.70	0.8738	-0.4400	0.6766

Next, the proposed method is applied to the same system. In this case, vector fitting is first applied to extract the poles and residues of the system from the S-parameters. An order of approximation of 18 is used and the constant and proportional terms are set to zero. All the poles a_n and residues c_n and their corresponding eye heights E_h and eye widths E_w are then stored in the structures as shown in Eqs. (12) and (13) as the inputs and targets.

$$input, x = \begin{bmatrix} c_{1,11} & c_{1,12} & c_{1,13} & \cdots \\ c_{2,11} & c_{2,12} & c_{2,13} & \cdots \\ \vdots & \vdots & \vdots & \cdots \\ c_{N,11} & c_{N,12} & c_{N,13} & \cdots \\ c_{1,21} & c_{1,22} & c_{1,23} & \cdots \\ c_{2,21} & c_{2,22} & c_{2,23} & \cdots \\ \vdots & \vdots & \vdots & \cdots \\ a_{1,1} & a_{1,2} & a_{1,3} & \cdots \\ a_{2,1} & a_{2,2} & a_{2,3} & \cdots \\ \vdots & \vdots & \vdots & \cdots \\ a_{N,1} & a_{N,2} & a_{N,3} & \cdots \end{bmatrix} \quad (12)$$

$$target, t = \begin{bmatrix} E_{h,1} & E_{h,2} & E_{h,3} & \cdots \\ E_{w,1} & E_{w,2} & E_{w,3} & \cdots \end{bmatrix} \quad (13)$$

where $E_{h,j}$ are the eye-heights and $E_{w,j}$ are the eye-widths. Since a_n and c_n are complex, they have real and imaginary parts and are separated as shown in Eqs. (14) and (15).

$$c_{n,mj} = \begin{bmatrix} re(c_{n,mj}) \\ im(c_{n,mj}) \end{bmatrix} \quad (14)$$

$$a_{n,kj} = \begin{bmatrix} re(a_{n,j}) \\ im(a_{n,j}) \end{bmatrix} \quad (15)$$

Then, multiple ANN networks with varying number of hidden neurons are trained to model the system, and the performance of each network is investigated. The results are tabulated in Table 2. We see that the proposed method speeds up the training process significantly due to a reduction in the number of inputs, while maintaining good accuracy. Since a vector fitting of order 18 was used, and the same poles were used for S_{11} and S_{12}, the total number of inputs was reduced from 2004 real valued numbers in the previous case to only 108 in this case. This speeds up the training and evaluation process significantly which proves the viability of the proposed method. For example, for the case of 4 hidden neurons, the proposed method requires only 0.35 s to train the network compared to 199.86 s in the conventional method. This is a speed-up of over 571×.

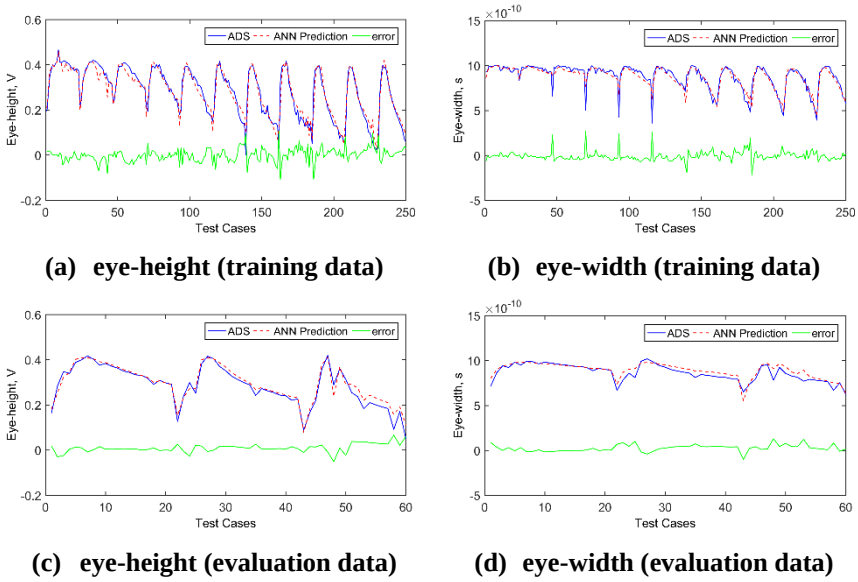
Table 2. Details of all neural networks for Case 1 (proposed method).

No. of Hidden Neurons	No. of Trained Models	Total Training Time (s)	Training Time per Model (s)	Average Training Perfor-mance	Average Evaluation Perfor-mance	Best Overall Perfor-mance
1	100	22.97	0.23	0.6400	0.5960	0.8785
2	100	23.82	0.24	0.6718	0.4795	0.8737
3	100	28.61	0.29	0.7204	0.5649	0.8912
4	100	35.04	0.35	0.7218	0.5802	0.9100
5	100	50.04	0.50	0.7483	0.6808	0.8789
6	100	63.11	0.63	0.7337	0.6648	0.8803
7	100	83.90	0.84	0.7681	0.6912	0.8829
8	100	103.65	1.04	0.7511	0.6292	0.8834
9	100	117.87	1.18	0.7471	0.6295	0.8799
10	100	391.40	3.91	0.7379	0.5805	0.8546

To further investigate the accuracy of the proposed method, the output eye-height and eye-width for one of the ANN networks with hidden neurons equal to 4 is selected and compared to the benchmark result from ADS. The result for both the training data and evaluation data are shown in Fig. 5 where the test cases are plotted with increasing electrical lengths, *E* and characteristic impedances, *Z*. We see that the trained ANN is able to predict the eye-height and eye-width with minimal error.

Figure 6 shows the plot of the training error, validation error, and testing error versus epochs/iterations of the selected neural network model which shows the errors decreasing with each iteration until training was terminated. Figure 7 shows the error histogram of the training, validation, and testing samples after the training was completed. The Levenberg-Marquardt method is known to be very memory

intensive when the training set is large or when the total number of weights and biases of the neural model is large, because excessive memory will be used for the Jacobian matrix storage and multiplication [19]. The update rule for the Levenberg-Marquardt algorithm can be found in [20]. Table 3 compares the memory needed to store the Jacobian matrices for the different number of hidden neurons. It is shown that our proposed method needs approximately 18 times less memory for the Jacobian matrix storage compared to the conventional method.



(a) eye-height (training data) **(b) eye-width (training data)**

(c) eye-height (evaluation data) **(d) eye-width (evaluation data)**

Fig. 5. Comparison between simulated and predicted result for Case 1.

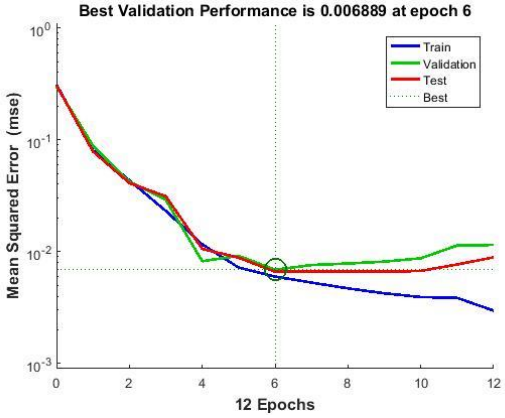


Fig. 6. Training error, validation error, and testing error versus epochs for Case 1.

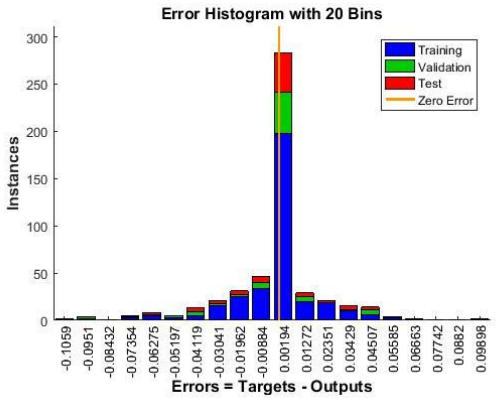


Fig. 7. Error histogram after training for Case 1.

Table 3. Memory used to store Jacobian Matrix (Case 1).

No. of Hidden Neurons	Memory Needed to Store Jacobian Matrix	
	Conventional Method (byte)	Proposed Method (byte)
1	8,116,240	457,424
2	16,224,384	906,752
3	24,332,528	1,356,080
4	32,440,672	1,805,408
5	40,548,816	2,254,736
6	48,656,960	2,704,064
7	56,765,104	3,153,392
8	64,873,248	3,602,720
9	72,981,392	4,052,048
10	81,089,536	4,501,376

3.2. Case 2: Lossy transmission line

Next, the example is repeated for the case of a lossy transmission line. The simulation setup is shown in Fig. 8. In this case, the characteristic impedance *Z*, physical length *L*, and attenuation, *A* are varied. The values of *Z* used are 30 Ω , 40 Ω, 45 Ω , 55 Ω , 75 Ω, 90 Ω, 120 Ω, 150 Ω, 180 Ω, and 210 Ω. The values of *L* are varied from 25 mm to 100 mm with a uniform step of 25 mm while the values of *A* are varied between 0.0001 dB m^{-1}, 0.01 dB m^{-1}, 1 dB m^{-1}, 10 dB m^{-1}, and 100 dB m^{-1} in order to cover a wide range of design space. Similar to Case 1, vector fitting of order 18 was used, resulting in 108 inputs.

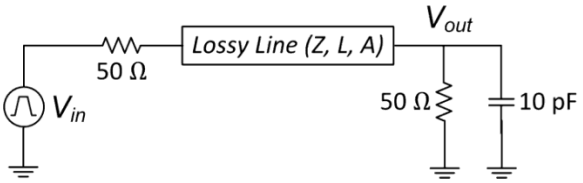


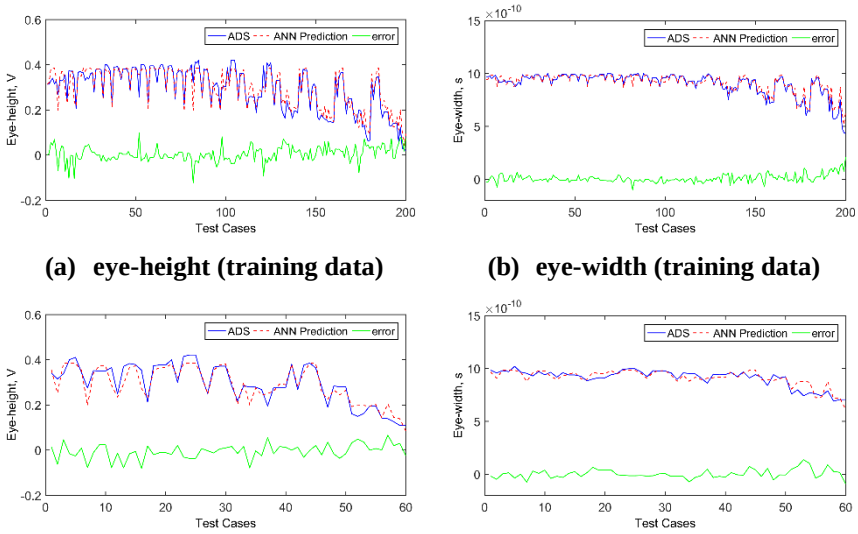
Fig. 8. Circuit in Case 2.

Similar to the previous case, ADS is again used to simulate the S-parameters for the ANN training process and also to simulate the eye diagram and the corresponding eye-heights and eye-widths which are used as the outputs. Vector fitting is used to extract the poles and residues with an order of 18 and these are used as the inputs to the ANN as in Eqs. (12)-(15). Results for different neural networks are shown in Table 4. We see again that the networks are able to provide accurate results while retaining a very fast training time.

Figure 9 shows the comparison between the output eye-height and eye-width from the ANN with the benchmark result from ADS for both the training and evaluation data for a network with two hidden neurons. The test cases in the plots are arranged with increasing *A*, *L* and *Z*.

Table 4. Details of all neural networks for Case 2 (proposed method).

No. of Hidden Neurons	No. of Trained Models	Total Training Time (s)	Training Time per Model (s)	Average Training Perfor-mance	Average Evaluation Perfor-mance	Best Overall Perfor-mance
1	100	20.03	0.20	0.5949	0.5320	0.7931
2	100	21.13	0.21	0.6087	0.4565	0.8579
3	100	25.89	0.26	0.6450	0.5423	0.8132
4	100	32.41	0.32	0.6355	0.4229	0.7992
5	100	39.57	0.39	0.6893	0.4967	0.8435
6	100	50.74	0.51	0.6822	0.3051	0.8270
7	100	64.93	0.65	0.6809	0.4331	0.8515
8	100	79.46	0.79	0.7012	0.3973	0.8196
9	100	95.13	0.95	0.6279	0.3040	0.8491
10	100	104.81	1.05	0.6743	0.3952	0.8081



(a) eye-height (training data) **(b) eye-width (training data)**

(c) eye-height (evaluation data) **(d) eye-width (evaluation data**

Fig. 9. Comparison between simulated and predicted result for Case 2.

4. Conclusion

In this work, vector fitting is used to disassemble the S-parameters into poles and residues, and then the obtained poles and residues are used as the inputs for an artificial neural network for eye-height and eye-width prediction. This approach reduces the number of inputs into the neural network which speeds up the training process significantly, compared to using the S-parameters directly. Future work will focus on an in-depth analysis and comparisons between the proposed method and other dimensionality reduction method along with the applications of these methods on more advanced circuit configurations such as in a complete high speed link.

Acknowledgments

This work was supported by the Malaysian Ministry of Higher Education under the FRGS grant no. 203/PELECT/6071246.

References

1. Fan, J.; Ye, X.; Kim, J.; Archambeault, B.; and Orlandi, A. (2010). Signal integrity design for high-speed digital circuits: Progress and directions. *IEEE Transactions on Electromagnetic Compatibility*, 52(2), 392-400.
2. Zhang, Q.J.; Gupta, K.C.; and Devabhaktuni, V.K. (2003). Artificial neural networks for RF and microwave design- From theory to practice. *IEEE Transactions on Microwave Theory and Techniques*, 51(4), 1339-1350.
3. Beyene, W.T. (2007). Application of artificial neural networks to statistical analysis and nonlinear modeling of high-speed interconnect systems. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 26(1), 166-176.
4. Liu, B.; Zhao, D.; Reynaert, P.; and Gielen, G.G. (2011). Synthesis of integrated passive components for high-frequency RF ICs based on evolutionary computation and machine learning techniques. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 30(10), 1458-1468.
5. Sadrossadat, S.A.; Cao, Y.; and Zhang, Q.J. (2013). Parametric modeling of microwave passive components using sensitivity-analysis-based adjoint neural-network technique. *IEEE Transactions on Microwave Theory and Techniques*, 61(5), 1733-1747.
6. Na, W.C.; and Zhang, Q.J. (2014). Automated knowledge-based neural network modeling for microwave applications. *IEEE Microwave and Wireless Components Letters*, 24(7), 499-501.
7. Ambasana, N.; Gope, D.; Mutnury, B.; and Anand, G. (2014). Application of artificial neural networks for eye-height/width prediction from S-parameters. *Proceedings of IEEE 23rd Conference on Electrical Performance of Electronic Packaging and Systems*. Portland, Oregon, 99-102.
8. Cao, Y.; Wang, G.; and Zhang, Q.J. (2009). A new training approach for parametric modeling of microwave passive components using combined neural networks and transfer functions. *IEEE Transactions on Microwave Theory and Techniques*, 57(11), 2727-2742.

9. Zhang, W.; Feng, F.; Zhang, J.; Zhang, S.; Gongal-Reddy, V.; and Zhang, Q.J. (2016). Advanced parametric modeling using neuro-transfer function for EM based multiphysics analysis of microwave passive components. *IEEE MTT-S International Microwave Symposium (IMS)*. San Francisco, California, 1-3.
10. Feng, F.; Gongal-Reddy, V.; Zhang, C.; Ma, J.; and Zhang, Q.J. (2017). Parametric modeling of microwave components using adjoint neural networks and pole-residue transfer functions with EM sensitivity analysis. *IEEE Transactions on Microwave Theory and Techniques*, 65(6), 1955-1975.
11. Zhang, Q.J.; and Gupta, K.C. (2000). *Neural networks for RF and microwave design*. Norwood, MA: Artech House.
12. Wang, F.; Devabhaktuni, V.K.; Xi, C.; and Zhang, Q.J. (1999). Neural network structures and training algorithms for RF and microwave applications. *International Journal of RF and Microwave Computer-Aided Engineering*, 9(3), 216-240.
13. Gustavsen, B.; and Semlyen, A. (1999). Rational approximation of frequency domain responses by vector fitting. *IEEE Transactions on Power Delivery*, 14(3), 1052-1061.
14. Gustavsen, B. (2006). Improving the pole relocating properties of vector fitting. *IEEE Transactions on Power Delivery*, 21(3), 1587-1592.
15. Deschrijver, D.; Mrozowski, M.; Dhaene, T.; and De Zutter, D. (2008). Macromodeling of multiport systems using a fast implementation of the vector fitting method. *IEEE Microwave and Wireless Components Letters*, 18(6), 383-385.
16. Chinea, A.; and Grivet-Talocia, S. (2011). On the parallelization of vector fitting algorithms. *IEEE Transactions on Components, Packaging and Manufacturing Technology*, 1(11), 1761-1773.
17. Keysight Technologies. (2006). Advanced design systems (ADS). Retrieved June 8, 2016, from http://www.keysight.com
18. Nguyen, D.; and Widrow, B. (1990). Improving the learning speed of 2-layer neural networks by choosing initial values of the adaptive weights. *Proceedings of International Joint Conference on Neural Networks*. San Diego, California, 21-26.
19. Wilamowski, B.M.; and Yu, H. (2010). Improved computation for Levenberg-Marquardt training. *IEEE Transactions on Neural Networks*, 21(6), 930-937.
20. Hagan, M.T.; and Menhaj, M.B. (1994). Training feedforward networks with the Marquardt algorithm. *IEEE Transactions on Neural Networks*, 5(6), 989-993.