Journal of Electronics Manufacturing, Vol. 6, No. 2 (June, 1996) 79–91
© World Scientific Publishing Company

INTEGRATED NEURAL NETWORK MODELING FOR ELECTRONIC MANUFACTURING

MANISH MARWAH, YUAN LI and ROOP L. MAHAJAN
Center for Advanced Manufacturing and Packaging of Microwave, Optical and Digital Electronics,
Department of Mechanical Engineering, University of Colorado, Boulder CO 80309-0427, USA

Received 30 September 1995
Accepted 29 January 1996

This paper addresses issues involved in the modeling of electronic manufacturing processes for optimization and control using artificial neural networks (ANNs). A modeling methodology is presented which integrates a number of techniques to counter the commonly experienced problems of selecting the 'right' network structure, over-training and long training times in building economical and accurate ANN models. This methodology has been implemented as an automated user-friendly ANN modeling software — CU-ANN. The main features of our methodology are data pre-processing, 'simple to complex' network structure approach and simultaneous training and testing. The neural networks considered have feed forward architecture and use error back-propagation algorithm for training. We have successfully applied this ANN modeling methodology to a number of simulated and real-life electronic manufacturing problems. These include stencil printing and simulated wafer fab. process data. The results indicate that our approach produces accurate, economical models and can handle a wide variety of data sets.

1. Introduction

For control or optimization of a process, it is essential to develop a process model. If the process is highly complex and nonlinear (as most electronic manufacturing processes are), a physical model may be difficult to build. Generally, several assumptions are made which make the accuracy and utility of such models suspect. Empirical models such as statistical regression models are then built from the experimental data. These models, however, have their own limitations. They may require a large number of experimental data and may have poor predictive capabilities, especially when the number of input/output variables is large. Artificial neural networks (ANNs) offer an attractive alternate modeling technique. It has been shown that ANNs are good at approximating complex, nonlinear data. In fact, it has been proved that a multilayer feed forward neural network can approximate virtually any function to any degree of accuracy.[1] In addition, they have excellent predictive capabilities and can be configured to be self-adaptive. These characteristics have led to extensive use of ANNs in process modeling. The most commonly used ANNs are feed forward networks using backpropagation learning algorithm.[2]

Briefly, artificial neural networks are massively parallel, highly inter-connected systems of computational nodes or neurons. A typical ANN structure is shown in Fig. 1. It consists of an input layer into which the input vector (independent variables) is fed in, as output layer which produces the output vector (dependent variables), and one or several hidden layers in between. The hidden layers link the input and output layers together and allow for complex, nonlinear interactions between the inputs to produce the desired outputs. Computations are done in the hidden layers and the output layer, but not the input layer.

Consider a multiple input–output relation mapped by an ANN. It can be mathematically expressed as a recursive formula relating the output of a layer to its input

$$Y_i^l = F\left(\sum_{j=1}^{N_{l-1}} w_{ij}^l Y_j^{l-1} + b_i^l\right), \qquad (1)$$

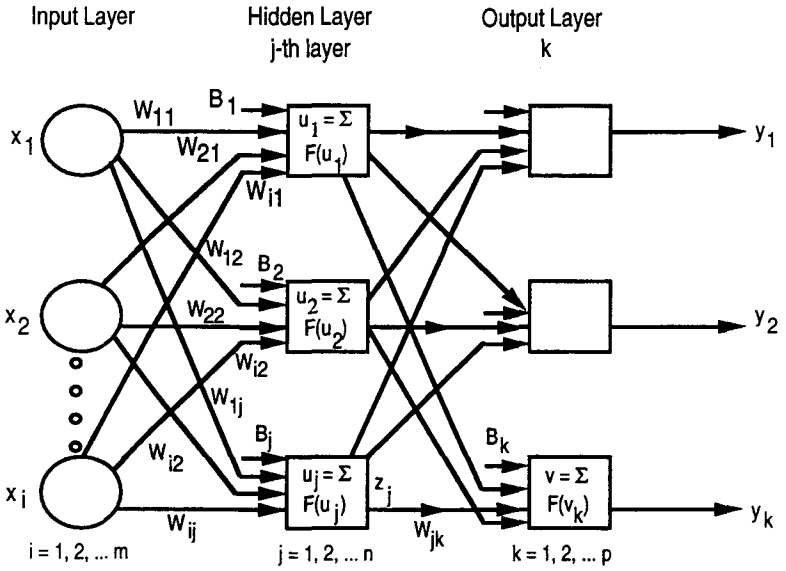

Fig. 1. An artificial neural network (ANN) structure.

where Y_i^l is the output of the ith neuron in the lth layer, w_{ij}^l is the weight on the connection from the jth neuron in the $(l-1)$th layer to the ith neuron in the lth layer, b_i^l is the bias connected to the ith neuron in the lth layer and N_{l-1} is the number of neurons in the $(l-1)$th layer. Note that $Y_i^0 = X_i$, where X_i is the input, and N_0 is the number of inputs. F is the activation function which may be thought of as providing a nonlinear gain for the artificial neuron. It is typically a sigmoid function, Eq. (2), and bounds the output from any neuron in the network

$$F(u) = \frac{1}{(1+e^{-u})}. \tag{2}$$

To train an ANN for developing a process model, a set of input–output data is required which should be representative of the process. The first step of training is the forward pass, which consists of calculating the output vector by running the input vector through the network. This is followed by a backward pass, where the error derivatives are calculated for each weight. The error derivatives for a weight are summed until all the training data points have been run through the network once. This constitutes an epoch. The weights are updated after each epoch such that the network error decreases. Readers are referred to Refs. 3 and 4 for further details.

The training process, briefly outlined above, can, however, be difficult and quite time-consuming. Training mainly depends on the network structure, transformation of data and the search algorithm used. The choice of these factors plays a crucial role in the ANN model development.

The network structure has a significant influence on the predictive capability of the network.[5] Several studies have addressed the issue of optimal network structure. Kim and May[6] use statistical experimental design to determine an optimal network for a specific application. Bhat and McAvoy[7] propose a stripping algorithm, starting with a large network and then reducing the network complexity by removing unnecessary weights/nodes. This 'complex to simple' procedure requires heavy and tedious computation. Villier and Barnard[8] conclude that although there is no significant difference between the optimal performance of one or two hidden layer networks, single layer networks do better classification on average.

A few authors advocate periodic testing while training,[4,9] so that the network does not overfit the data. Furthermore, they suggest that training be stopped whenever the test mean squared error reaches a minima. It has been shown in Ref. 10 that this approach may not always lead to the optimum network.

Pre-processing of data can lead to substantial improvements in the training process. Not much attention has been given to this aspect of training. Kwon *et al.*[11] propose a data pre-processing algorithm for a highly skewed data set.

Another active area of research involves devising faster search algorithms. Several variations of plain back-propagation (i.e. back-propagation with

steepest gradient) have been proposed. A good comparison of the various techniques for optimizing backpropogation is given in Ref. 12. We deploy both steepest gradient and adaptive steepest gradient[13] search algorithms.

In this paper, we present a comprehensive methodology which integrates a "simple to complex" network structure approach, data pre-processing and simultaneous training and testing. The next section describes our training methodology. Before applying our methodology to real electronic manufacturing problems, we tested it on a wide variety of mathematical functions. These tests, along with ANN models of stencil printing process in a surface mount assembly line and simulated wafer fab. data, are described in Sec. 3. Finally, a few concluding remarks are given in Sec. 4.

2. Our ANN Modeling Methodology

An overview of our modeling methodology is shown as a flowchart in Fig. 2. Some of the details have been left out to simplify the flowchart. This methodology has been implemented as a user-friendly software,

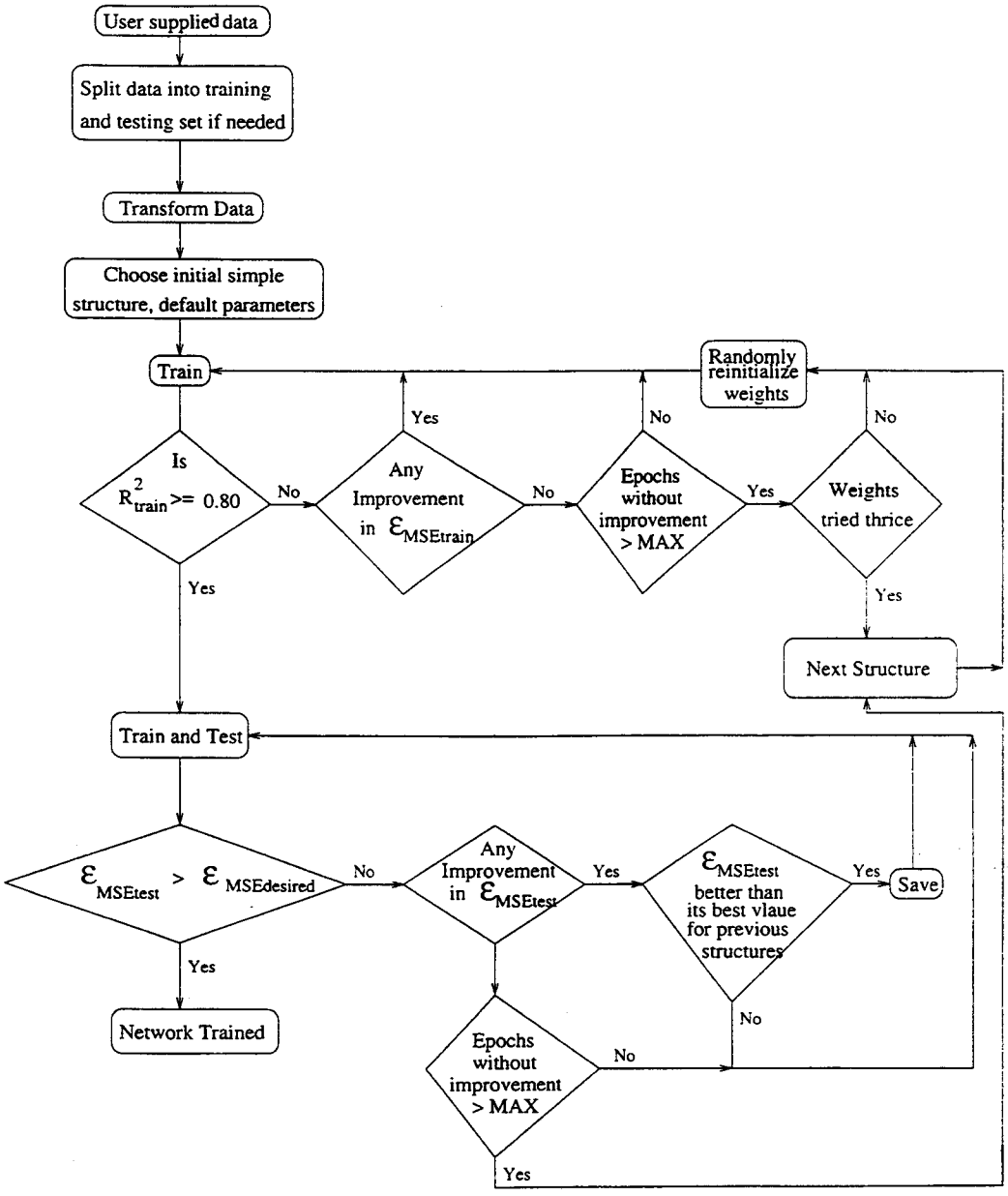

Fig. 2. Flow chart showing our neural network training methodology.

called CU-ANN, on an IBM-PC, and is fully described in Ref. 10. Here, we briefly describe the three important elements mentioned above. To facilitate discussion, performance measures are discussed first.

2.1. *Performance measures*

In order to gauge the performance of a network on a data set, some quantitative measures are required. If it is a classification task, the most common and natural measure is the percentage of examples correctly classified by the network. However, for modeling continuous output, composite measures are needed.

We use three measures to gauge the network performance. These are percentage relative error, $\bar{\mathcal{E}}_{\text{Rel}}$, average mean squared error $\bar{\mathcal{E}}_{\text{MSE}}$, and the correlation coefficient, R^2. They are defined as

$$\bar{\mathcal{E}}_{\text{MSE}} = \frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2\,, \tag{3}$$

$$\bar{\mathcal{E}}_{\text{Rel}} = \frac{1}{N}\sum_{i=1}^{N}\left|\frac{\hat{y}_i - y_i}{y_i}\right|\,, \tag{4}$$

$$R^2 = 1 - \frac{\sum_{i=1}^{N}(\hat{y}_i - y_i)^2}{\sum_{i=1}^{N}(y_i - \bar{y})^2}\,, \tag{5}$$

where

N is the total number of data points,

$\hat{y}_i$ is the predicted output,

y_i is the actual output, and

$\bar{y}$ is the actual output mean.

All of these measures have drawbacks and are not good under all situations. However, examining all three gives a good idea of how the network performs.

$\bar{\mathcal{E}}_{\text{MSE}}$ is the absolute error in the sense that its units are square of the units of the output. Without comparing $\bar{\mathcal{E}}_{\text{MSE}}$ to the actual output, one cannot determine the goodness of an ANN model. When the output is small, even a low $\bar{\mathcal{E}}_{\text{MSE}}$ can correspond to a high relative error. The other two measures, R^2 and $\bar{\mathcal{E}}_{\text{Rel}}$, are relative. $R^2 = 1$ and $\bar{\mathcal{E}}_{\text{Rel}} = 0$ indicate a perfect fit. Note, however, that the value of R^2 depends not only on the goodness of fit but also on the distribution of the data points. If the data points cluster together, the denominator in Eq. (5) would have a small value. This would lead to a small R^2 even when the numerator is small, which indicates a good fit. Similarly, a high $\bar{\mathcal{E}}_{\text{Rel}}$ value may not always mean that a model is bad. Inaccurate prediction of just a few data points with low output values can result in a large $\bar{\mathcal{E}}_{\text{Rel}}$. Thus, a bad R^2 or $\bar{\mathcal{E}}_{\text{Rel}}$ alone does not necessarily imply a poor ANN model.

2.2. *Data and pre-processing*

In our approach for developing an ANN model, we use two sets of data. One for training the network and the other for testing it. It is generally desirable if some additional data points are available for validating the final model. For the purpose of selecting data for model-building, we distinguish between two situations: 1) Enough process data is not available (e.g. the process is new) and the ANN model developer can specify the data required; 2) The historical process data is already available but obtaining any

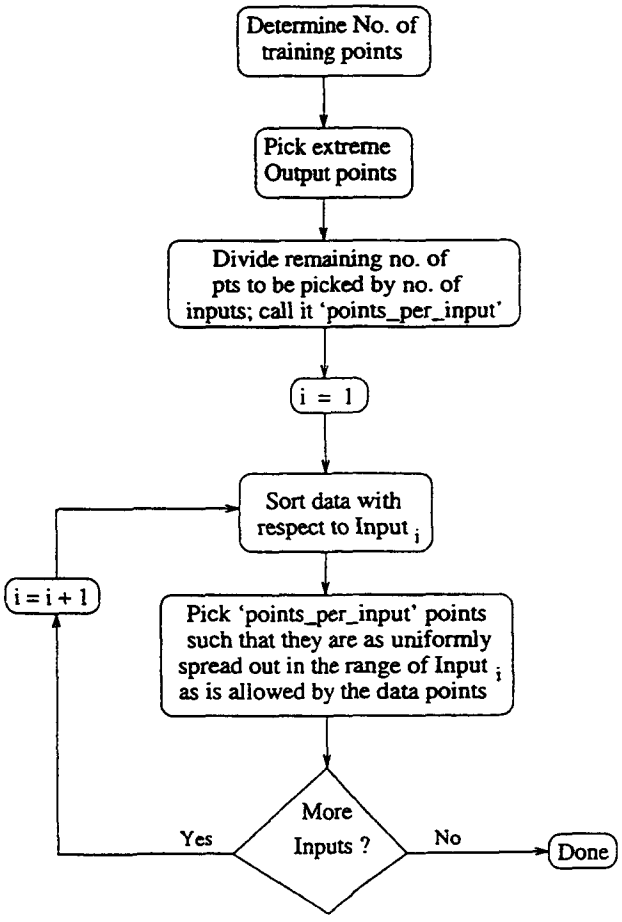

Fig. 3. Flow chart showing the algorithm used to split a data set into a training and a test data sets.

more data points desired by the model developer is difficult and expensive.

In the first case, we use the design of experiment (DOE) approach to conduct carefully designed experiments over the entire input domain such that the general behavior of the process can be captured. The data points so generated are used to train the model. Additional data points, when necessary, are gathered to serve as testing points.

In the second situation, when the data is already available, it is split into a training set and a testing set. Typically, three quarters of the data is used for training. The remainder is used for testing the ANN model. The training points are chosen from the data set such that they are uniformly spread out over the input domain space. The extremes of the output are also included in the training set. The data points that remain after the training points have been chosen are used for testing. Figure 3 shows the algorithm used to pick out the training points from a data set.

The inputs are linearly normalized between 0 and 1. For the outputs, the normalization range is between 0.2 and 0.8 so that the output neurons operate in the relatively linear portion of the sigmoid. If the range of an input or output is large (the minimum and maximum values differ by orders of magnitude), small variations in its value are lost during linear normalization. Therefore, such output or input values are logarithmically normalized before training.

2.3. *Selecting network structure*

Successful training of an ANN model is highly dependent on the structure of the neural network used. A network with fewer hidden neurons than required would not be able to model the complexity of the process, whereas one with excessive neurons may overfit the data. To determine the appropriate network, we follow a 'simple to complex' network structure approach. Initially, a simple network structure is chosen. It typically consists of two neurons in a single hidden layer. If learning is unacceptably slow, the training is stopped and the complexity of the network is increased by using an additional neuron. This is done only after trying a number of randomized initial weights (usually three) for that structure. The rate of decrease of mean squared error of the network, $\bar{\mathcal{E}}_{\text{MSE}}$ (Eq. (3)), is monitored to determine if learning is unacceptably slow. In most of the cases, one hidden layer is sufficient, although we found some examples where a two hidden layer network performs better.

In our software, CU-ANN, the process of changing the network structure has been automated. In the 'auto' mode, when the rate of decrease of $\bar{\mathcal{E}}_{\text{MSEtest}}$ becomes lower than a user specified value, the software augments the network structure by a neuron, and training begins anew. If the best performance of a network structure is better than all the previous structures tried, then the weights are saved in a file. The network performance measure, mean squared error, $\bar{\mathcal{E}}_{\text{MSE}}$, is also saved. We have found this to be a very useful method of identifying the optimal network structure for a data set. One can load the data and run the software in 'auto' mode for a few hours (depending on the number of data points) without the requirement of any human intervention. The software saves the best network structure for the data.

2.4. *Simultaneous training and testing*

Before training begins, all the network weights are randomly initialized to small values around zero. The weights are updated after each epoch, where one epoch comprises running all the training points through the network. After each epoch, the R-squared and the mean squared error of the network are computed. These are the R^2_{train} and $\bar{\mathcal{E}}_{\text{MSEtrain}}$ as they just involve the training points. Testing involves running the test data set through the network; however, the weights are not updated. Simultaneous training and testing is not performed until R^2_{train} reaches some threshold value (0.8). Below that value, the model has poor fidelity and is not worth testing. During simultaneous training and testing, the network test measures, R^2_{test} and $\bar{\mathcal{E}}_{\text{MSEtest}}$, are calculated after each epoch. The performance of the network on the test data set reflects its predictive capabilities. The $\bar{\mathcal{E}}_{\text{MSEtest}}$ is compared against the desired $\bar{\mathcal{E}}_{\text{MSE}}$ of the model, $\bar{\mathcal{E}}_{\text{MSEdesired}}$, to determine if the network has been trained.

$\bar{\mathcal{E}}_{\text{MSEtrain}}$ always goes down as training proceeds. This follows from the very definition of training. The weights are updated such that the network error decreases. $\bar{\mathcal{E}}_{\text{MSEtest}}$, on the other hand, need not always decrease. In fact, it has been observed that it usually decreases first and then begins to rise. In some cases, however, it decreases again. An increasing $\bar{\mathcal{E}}_{\text{MSEtest}}$ signals overfitting of data, which implies a poor prediction. An over-trained model may fit the training points very well, but it does

not well represent the real process model. In such a case, $\bar{\mathcal{E}}_{\text{MSEtrain}}$ is very low but $\bar{\mathcal{E}}_{\text{MSEtest}}$ is high. If the network completely overfits (passes through all the training points, i.e. $\bar{\mathcal{E}}_{\text{MSEtrain}} \approx 0$, but $\bar{\mathcal{E}}_{\text{MSEtest}}$ is high), learning would stop as there is no driving error to change the weights. When the overfit is partial, the weights can change such that the situation is reversed and the ANN model again moves closer to the real model. In this situation the $\bar{\mathcal{E}}_{\text{MSEtest}}$ curve would rise and then fall again.

In CU-ANN, whenever $\bar{\mathcal{E}}_{\text{MSEtest}}$ decreases, the weights are stored. Thus even if the $\bar{\mathcal{E}}_{\text{MSEtest}}$ curve rises, the weights at the lowest point are not lost. After $\bar{\mathcal{E}}_{\text{MSEtest}}$ starts rising, the software continues training for some pre-defined number of epochs to see if $\bar{\mathcal{E}}_{\text{MSEtest}}$ shows any signs of falling. If $\bar{\mathcal{E}}_{\text{MSEtest}}$ continues rising, it switches to the next network structure or the same structure with another set of randomly initialized weights.

3. Applications

In this section we present applications of our modeling approach to three different cases: i) mathematical functions, ii) surface mount stencil printing process and iii) wafer fabrication simulation data. The purpose of modeling mathematical functions is to benchmark the approach. The other two processes are representative of two typical microelectronics manufacturing processes. In (ii), experiments are expensive to conduct and we use the design of experiment (DOE) approach to generate data points to build the model. In (iii), the data can be simulated using mathematical models. However, the output variable's range is large and requires special data pre-processing.

3.1. *Mathematical functions*

3.1.1. *Functions without noise*

The following two input variable functions were tested: i) linear function, ii) second-order polynomial, iii) periodical function, iv) rational function, v) discontinuous function, vi) a function that is linear in x_1 direction and periodical in x_2 direction and vii) a function periodical in both directions. These functions represent processes of increasing complexity and are mathematically described as:

$$y = 2x_1 + 3x_2\,, \tag{6}$$

$$y = 5x_1 + 3x_2 - 10x_1x_2 + 5x_1^2 + 3x_2^2\,, \tag{7}$$

$$y = \sin(2x_1 + 4x_2)\,, \tag{8}$$

$$y = \frac{2x_1 + 3x_2}{0.1 + 5x_1^2 + 4x_2^2}\,, \tag{9}$$

$$y = \begin{cases} 0, & \text{if } x_1^2 + x_2^2 \le 0.5 \\ 1, & \text{otherwise}\,, \end{cases} \tag{10}$$

$$y = (x_1 + 1)\cos(6x_2 + 6)\,, \tag{11}$$

$$y = \sin(6x_1 + 7)\cos(6x_2 + 6)\,. \tag{12}$$

The region studied for each function was $x_1 \in [-1.0, 1.0]$ and $x_2 \in [-1.0, 1.0]$. The least number of data points possible were used for modeling each function. The data points were evenly distributed across the 2-dimensional input space.

Using the approach outlined in Sec. 2, ANN models with the simplest architecture that provided the best predicted capability were built. The modeling details are listed in Table 1.

As shown in column 6 of Table 1, the R^2_{test} values for most functions are above 90%, which

Table 1. ANN models for 'clean' functions.

Function Type	No. of Points		ANN Architecture	Epochs	R^2_{train} (%)	R^2_{test} (%)	$\bar{\mathcal{E}}_{\text{Rel}}$ (%)	$R^2_{\text{validation}}$ (%)	$\bar{\mathcal{E}}_{\text{Rel}}$ (%)
	Train	Test							
1	4	2	2-5-1	84	98	100	0.9	99.5	3
2	9	3	2-7-1	4433	99.9	91	25	97	25
3	16	5	2-5-1	25233	99.9	92	24	93	27
4	25	8	2-8-1	81586	99.9	90	28	88	30
5	49	15	2-5-5-1	8993	98	92	90	88	7
6	36	12	2-5-5-1	14759	99	96	58	98	58
7	72	13	2-10-10-1	114376	98	86	33	90	100

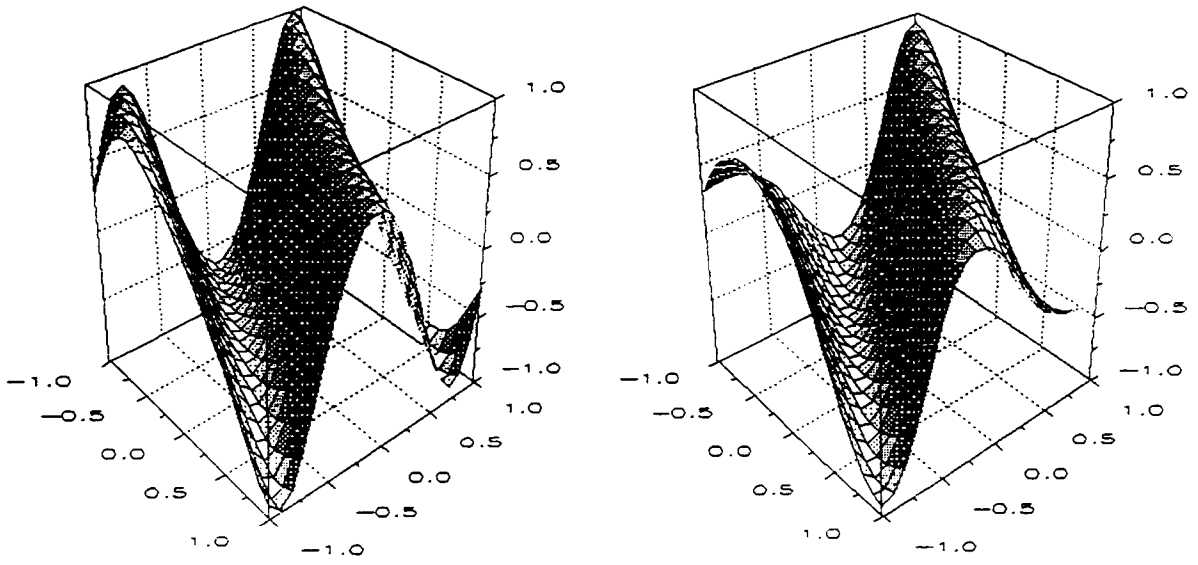

Fig. 4. (a) Surface plot of function 3, $\sin(2x_1 + 4x_2)$; (b) ANN model plot.

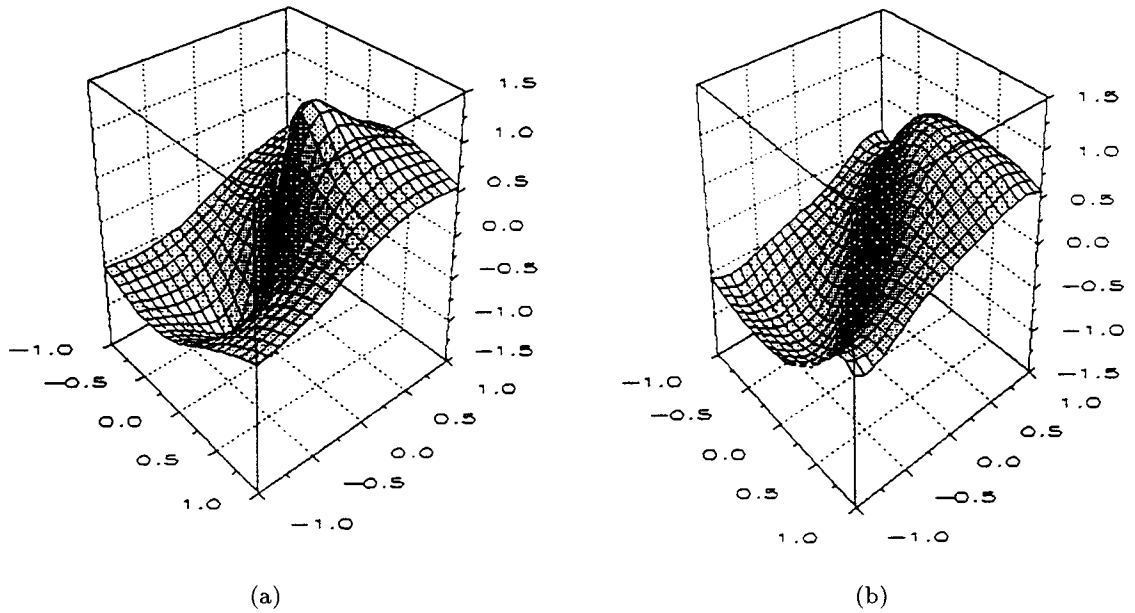

Fig. 5. (a) Surface plot of function 4, $\frac{2x_1+3x_2}{0.1+5x_1^2+4x_2^2}$; (b) ANN model plot.

indicates good accuracy of the ANN models. For further verification, the functions were used to generate an additional 441 points (21×21). These were used to validate the ANN models. The results are listed in column 7 of Table 1. They clearly indicate high accuracy of the models. Figures 4 to 8 show plots of the functions described by Eqs. (8) to (12), along with the predictions of the ANN models (for Eqs. (6) and (7), the plots are not shown here to save space. The match, however, was excellent). As shown in Figs. 4 and 5, functions given by Eqs. (8) and (9) are very nonlinear, with sinusoidal and rational characteristics. Function Eq. (10) introduces discontinuity (see Fig. 6), and functions given by (11)

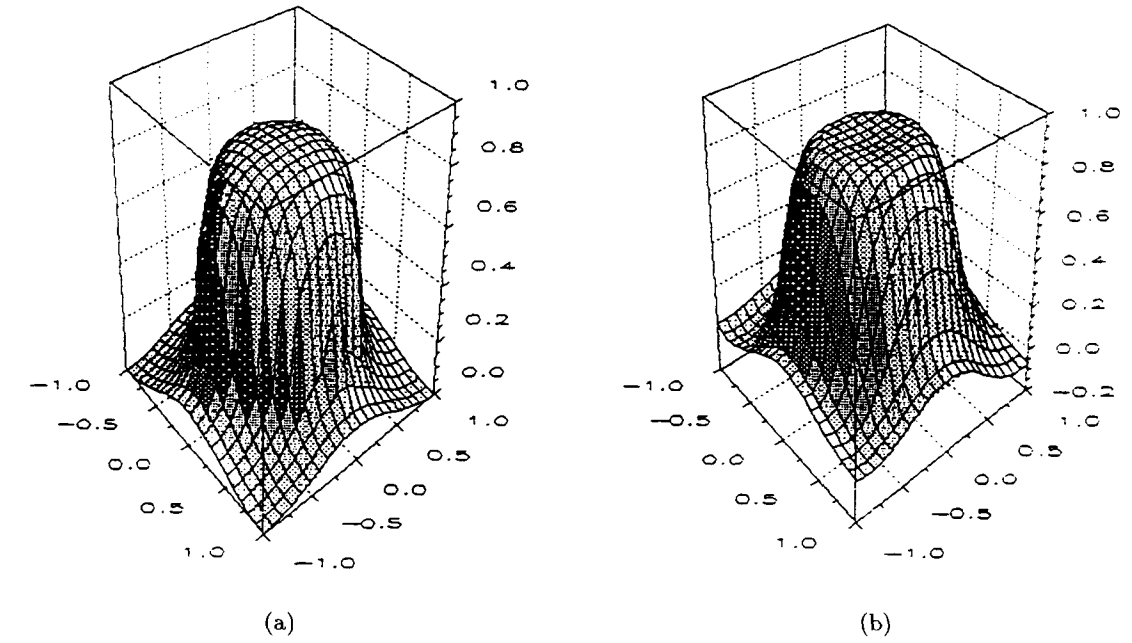

Fig. 6. (a) Surface plot of function 5, $y = 0$, if $x_1^2 + x_2^2 \leq 0.5$, $y = 1$, otherwise; (b) ANN model plot.

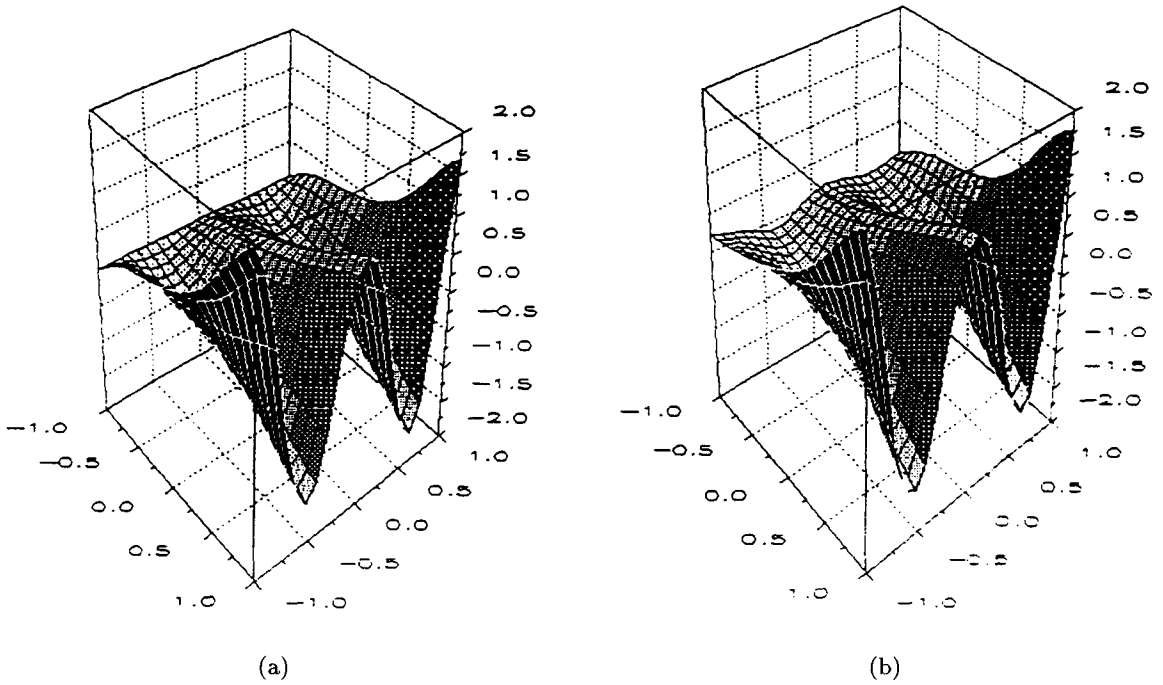

Fig. 7. (a) Surface plot of function 6, $(x + 1)\cos(6x_2 + 6)$; (b) ANN model plot.

and (12) depict multi-modal behaviors (see Figs. 7 and 8). Considering the small number of points used to develop these models (see column 2 of Table 1), the ANN model's predictive capability for these non-linear functions (see Figs. 4(b), 5(b), 6(b), 7(b) and 8(b)) is impressive.

3.1.2. *Functions with noise*

The ANN modeling approach was also tested using the above functions in the presence of noise. The objective was to understand the effect of noise on the model's development and accuracy. The noisy

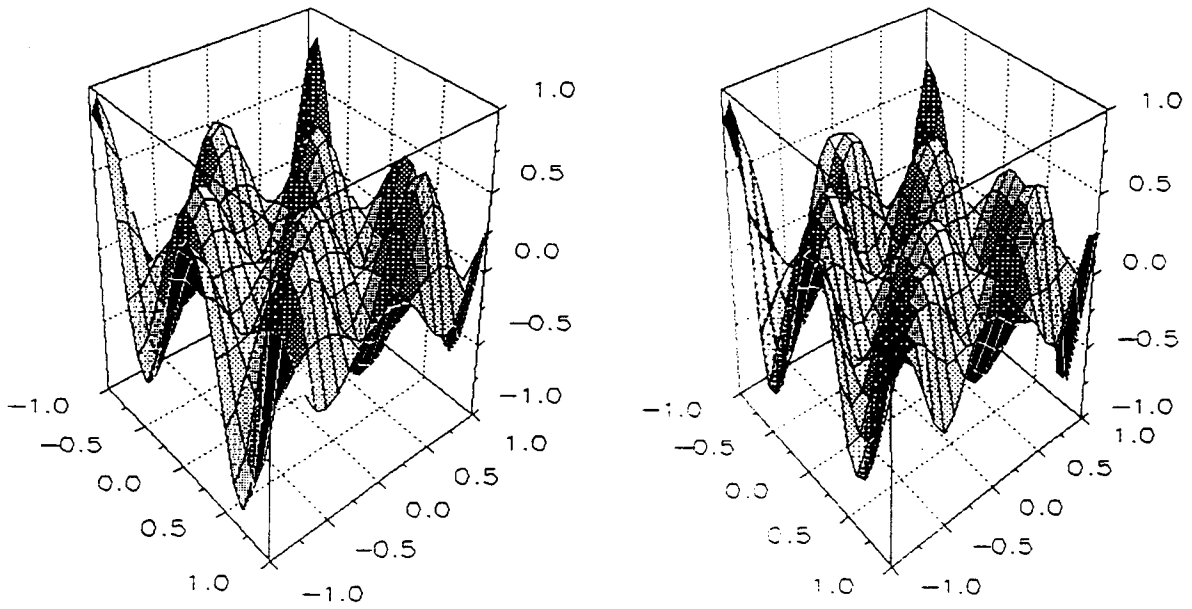

Fig. 8. (a) Surface plot of function 7, $\sin(6x_1 + 7)\cos(6x_2 + 6)$; (b) ANN model plot.

Table 2. ANN models for functions with 20% noise.

Function Type	No. of Points		ANN Architecture	Epochs	R^2_{train} (%)	R^2_{test} (%)	$\bar{\mathcal{E}}_{\text{Rel}}$ (%)	$R^2_{\text{validation}}$ (%)	$\bar{\mathcal{E}}_{\text{Rel}}$ (%)
	Train	Test							
1	4	2	2-5-1	83	91	99	2	92	15
2	9	3	2-7-1	6093	99.9	86	30	96	30
3	16	5	2-5-1	33171	99.9	87	28	93	29
4	25	8	2-8-1	28306	99	87	30	82	28
5	49	15	2-5-5-1	15703	91	84	15	82	17
6	36	12	2-5-5-1	94520	98	94	26	97	44
7	72	13	2-10-10-1	80521	97	80	49	86	100

data was generated using the formula

$$y_n = y \cdot (1.0 + \alpha \cdot rn), \tag{13}$$

where y is the clean data obtained from the functions, y_n is the corresponding noisy value, rn is a random number with 0 mean and variance of 1, and α is a measure of the degree of noise. In our computations, α was taken as 5%, 10% and 20%. The results are presented for $\alpha = 20\%$. Using the same number of data points as before, the ANN models were developed for these functions. The details are listed in Table 2.

A comparison of these results with those listed in Table 1 demonstrates the 'noise-filtering' capability of ANN models. This generalization capability is an attractive feature when applying to manufacturing processes which generally tend to be noisy.

3.2. *Stencil printing process*

In fine pitch surface mount assemblies, stencil printing is the first and perhaps the most critical step affecting the quality and yield of the whole assembly. Figure 9 shows a simple schematic of the process. In this study, we undertake the task of exploring the use of neural networks for modeling and optimization of fine pitch stencil printing process. The six design parameters listed in Table 3 were selected as the main effects requiring investigation. Here, squeegee

pressure was measured in number of screw turns used to adjust the pressure and squeegee angle was the angle between the squeegee and the vertical axis. The quality characteristic is the solder paste height. Of the six parameters, A and B are discrete variables and have only two levels. The other four are continuous variables and were assigned three levels. The lower and the upper bounds were chosen based on past experience and engineering judgment.[14]

Taguchi array L27 was selected to set up the experiment. Since the two and three-level factors were mixed together, the dummy treatment technique was used. For factors A and B, relatively less engineering information was available for level 1 of factor A and level 2 of factor B. Following standard practice,[15] these two levels were repeated using the dummy treatment technique.

Fifty four runs (27 for each direction of the printer) were made on FUJI GSP–II printer on one of the assembly lines of AT&T. The stencil used was of stainless steel with thickness stepped down to 8, 6 and 4 mil. The board material was FR–4. The solder paste's composition was 63Sn/37Pb with powder mesh size $-325/+500$. The metal load was 90% in weight percentage, and the paste viscosity was $1000 \pm 10\%$ kcps. A laser section microscope (LSM), with a measurement accuracy of 0.04 mil, was used to measure the solder paste height.

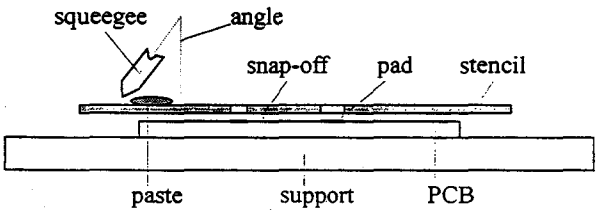

Fig. 9. A schematic of the stencil printing process.

The experimental trials were randomized. The solder paste height on the three 20-mil pitch pads, four 25-mil pitch pads and seven 50-mil pitch pads was measured on each board. The mean, $\bar{h}$, and the standard deviation (S) were computed for each pattern for all the twenty-seven trials using Eqs. 14 and 15:

$$\bar{h} = \frac{1}{N}\sum_{i=1}^{N} h_i\,, \tag{14}$$

$$S = \sqrt{\frac{\sum_{i=1}^{N}(h_i - \bar{h})^2}{N-1}}\,, \tag{15}$$

where h_i is the solder paste height at site i, and N is the number of sites measured for each pattern. Thus, a total of twenty-seven data points were obtained. These were then used to develop the neural network models. Two kinds of models were developed. They are described in detail below.

Table 3. Factors and their levels.

	Factors	Levels		
		1	2	3
A	squeegee material	metal	polyurethane	
B	pad material	HASL	bare copper	
C	snap-off distance (mil)	0	5	10
D	squeegee pressure (turn)	$\frac{1}{8}$	$\frac{1}{4}$	$\frac{3}{8}$
E	squeegee angle (deg)	18	22	26
F	squeegee speed (mm/s)	20	25	30

Table 4. ANN model for stencil printing process.

ANN Architecture	Epochs	Outputs	$R^2_{\text{train}}(\%)$	$\bar{\mathcal{E}}_{\text{Reltest}}(\%)$	$\bar{\mathcal{E}}_{\text{Relvalid}}(\%)$
line 6-4-3	4022	20-mil	97	4	6
		25-mil	96	4	5
		50-mil	98	5	14

3.2.1. *Mean model*

This model was developed to relate the mean solder paste height for each of the three pad patterns (20, 25 and 50 mil pitch) to the six design parameters. Using the approach outlined before, the twenty-seven data points were randomly divided into a training set with twenty points and a testing set with seven points. The optimum network structure was found to be 6-4-3 where the three numbers represent the six input neurons corresponding to the six design parameters, four hidden neurons and the three output neurons corresponding to the mean solder paste heights for the 20-mil, 25-mil and 50-mil pitch pads, respectively. The results are shown in Table 4. To further verify the fidelity of these models, thirteen new trials were later run on the production line. The relative errors for the three outputs were 6%, 5% and 14%, respectively (see column 6). Considering that there are six input variables and three outputs, the accuracy of the model developed using relatively few data points is excellent.

3.2.2. *Variation model*

An important objective of this investigation was to determine the optimum settings for minimum solder paste height variation. The average value of the standard deviations for the solder paste height for the three pad patterns (20, 25 and 50 mil pitch) was used as the output. Using the above modeling approach, the optimum network structure was found to be 6-3-1. The training R^2 was 95% and the testing relative error was 18%. Considering that only twenty-seven data points (twenty for training and seven for testing) were used to develop the model, the model presents reasonable predictive capability. Using the variation model in conjunction with a gradient descent algorithm, we determined the optimum settings for which the variation was minimum. For the optimized settings, the standard deviation was determined to be only 0.28 mil. This represents a factor of $\approx$ 2 improvement over the standard deviation of 0.50 mil under normal printing conditions.

Subsequently, three confirmation trials were conducted on the production line to verify the optimum settings. The results are shown in Table 5. The print quality and repeatability for the three trials were very good. The averaged standard deviation of the solder paste height remained almost constant for the three trials. The network prediction and the experiment results matched almost perfectly indicating accuracy of our modeling approach in a real manufacturing environment.

3.3. *Wafer fab. simulation data modeling*

In this last application considered, we develop ANN models for device performance simulations in a wafer fab. Input–output data for a device simulation can be generated using commercially available tools such as ATLAS. These simulations are generally accurate to around 2% average relative error, $\bar{\mathcal{E}}_{\text{Rel}}$. However, these device simulations can be computationally extensive and, therefore, time-consuming. As a result, computing the response for a particular input can be slow. ANN models, on the other hand, are extremely fast. However, they need to approach the accuracy

Table 5. Optimum settings and confirmation results.

Factors						Network Predicted Std. Deviation	Conformation Results (Trail No.)
A	B	C	D	E	F		
m	c	7	0.375	22	30	0.28	0.28 (1) 0.26 (2) 0.26 (3)

Table 6. Training statistics when a single global neural net is used.

No. of Data Points	$\bar{\mathcal{E}}_{\text{Rel}}$	Epochs	ANN Architecture	Time (min) on IBM PC 486
336	11%	24000	3-10-1	80

Table 7. Training statistics for device simulation data.

Sub-net	No. of Data Points	$\bar{\mathcal{E}}_{\text{Rel}}$	Epochs	ANN Architecture	Time (min) on IBM PC 486
1	65	2.63%	11000	3-8-1	9.17
2	32	3.01%	20000	3-11-1	12.33
3	38	1.71%	6400	3-5-1	2.66
4	37	3.1%	16400	3-9-1	4.00
5	54	1.55%	10000	3-6-1	6.16
6	184	1.83%	6000	3-8-1	12.00
				Total Time	46.32

of the device simulation tools. A potential hurdle in achieving this accuracy is the large range of the data output.

As a test case, we considered a typical device simulation data set consisting of 3 inputs (V_{gate}, V_{drain} and $V_{\text{substrate}}$), and 1 output (I_{drain}). The output, I_{drain}, varies from around 10^{-16} to 10^{-4}. A total of 336 data points was considered. The output was logarithmically transformed and a neural network was trained, following our approach outlined in Sec. 2. The results are summarized in Table 6. A 3-10-1 network provided the best results. An error of $\bar{\mathcal{E}}_{\text{Rel}} < 2\%$ was obtained in the transformed domain. However, when the output was transformed back to the primitive variables, by taking the antilog, the errors magnified. An $\bar{\mathcal{E}}_{\text{Rel}}$ of 11% was obtained. This is much higher than the desired value of 2% and hence was considered unacceptable.

In order to achieve greater accuracy, the input domain of the data was split into a number of overlapping sub-domains. Each of these was trained on a different network. The domain was split such that the range of the output over any sub-domain was significantly lower than the entire output range. The expectation was that a network could be trained faster and more accurately on a smaller range. The sub-domains overlapped so that the derivative of the model was continuous everywhere.

For the data set considered, the input domain was divided into 6 sub-domains. An overall model $\bar{\mathcal{E}}_{\text{Rel}}$ of 2.12% could be achieved (see Table 7). Furthermore, note that the total training time of all the six networks is only about 58% of the time taken by the single global network. For complete details of this methodology for developing overlapping sub-domain models, the reader is referred to Ref. 10.

4. Conclusions

In this paper, we have presented our ANN modeling techniques for application to electronic manufacturing. It is shown that integrating pre-processing techniques, 'simple to complex' network structure approach and simultaneous training and testing results in substantially efficient ANN models.

The benchmarking study using mathematical functions showed that the proposed ANN modeling approach is an accurate and efficient tool for modeling continuous valued simple as well as complex functions with relatively small number of data points. Furthermore, the models obtained from the data with random noise indicate that the models filter noisy data and capture the underlying functions.

We demonstrated application of our ANN modeling methodology to electronic manufacturing using stencil printing and wafer device simulation as test vehicles. In the former process which has four continuous and two discrete inputs, 27 data points were collected using Taguchi method. Based on these relatively few number of points, an ANN model with six inputs and three outputs was successfully developed using the proposed ANN modeling approach. This model, together with steepest gradient, was used to find the optimum settings for the process. The experimental results for this setting matched almost exactly with the ANN model's prediction.

In the second application, an ANN model was developed for wafer fab. simulation data which had a large range. It was shown that dividing up the input space into smaller regions and using a number of neural networks resulted in a model with the required high accuracy and lower training time.

Acknowledgment

This research was funded by SRC, grant 94-YJ-557. The authors gratefully acknowledge Silvaco for the wafer fab. data.

References

1. K. Hornik, M. Stinchcombe and H. White, "Multilayer feed forward networks are universal approximators", *Neural Networks* **2** (1989), 359–366.
2. D. E. Rumelhart, G. E. Hinton and R. J. Williams, "Learning internal representations by error propagation", *Parallel Distributed Processing*, vol. 1, MIT Press, Cambridge, MA, 1986, pp. 318–362,.
3. S. Haykin, *Neural Networks, A Comprehensive Foundation*, Macmillan, New York, NY, 1993.
4. M. Smith, *Neural Network for Statistical Modeling*, Van Nostrand Reinhold, New York, 1989.
5. E. Baum and D. Haussler, "What size net gives valid generalization?", *Neural Computation*, **1** (1989), pp. 151–160.
6. B. Kim and G. S. May, "An optimal neural network process model for plasma etching", *IEEE Trans. on Semiconductor Manufacturing* **7**(1) (1994).
7. N. V. Bhat and T. J. McAvoy, "Determining model structure for neural models by network stripping", *Computers and Chemical Eng.* **16**(4) (1992), 271.
8. J. Villiers and E. Barnard, "Backpropagation neural nets with one and two hidden layers", *IEEE Trans. on Neural Networks* **4**(1) (1992).
9. D. Hammerstrom, "Working with neural networks", *IEEE Spectrum* (July 1993).
10. M. Marwah, "Neural network modeling techniques for selected electronic manufacturing processes", MS Thesis, University of Colorado, Boulder, 1996.
11. T. M. Kwon, E. H. Feroz and H. Cheng, "Preprocessing of training set for backpropagation algorithm: histogram equalization", *IEEE Int. Conf. on Neural Networks*, 1994, Vol. 1, pp. 425–429.
12. E. Barnard and J. E. W. Holm, "A comparative study of optimization techniques for backpropagation", *Neurocomputing* **6**(19) (1994).
13. R. A. Jacobs, "Increased rates of convergence through learning rate adaption", *Neural Networks* **1**(1) (1988), 295–308.
14. Y. Li, R. L. Mahajan and N. Nikmanesh, "Fine pitch stencil printing process modeling and optimization", EPP-Vol. 10.1, *Advances in Electronic Packaging*, INTERPAC '95, pp. 119–127, *Journal of Electronic Packaging*, Vol. 118, No. 1, March 1996, 1–6. .
15. G. S. Peace, *Taguchi Methods: A Hands On Approach*, Addison Wesley, 1992.