Fault Diagnosis in Mixed-Signal Circuits via Neural-Network based classification Algorithms

Vivek Kumar Rajan, Krishna R. Pattipati and Jie Luo[*]

ABSTRACT

In this work, we implemented and evaluated several learning paradigms for general classification and diagnostic problems arising in mixed-signal circuit testing. The network models evaluated include Restricted Coulomb Energy (RCE) Neural Network, Learning Vector Quantization (LVQ), Decision Trees (C4.5), and Fuzzy Adaptive Resonance Theory (FuzzyArtmap). A Virtual Test Bench (VTB) was developed to pre-process and extract the fault-test dependency information used as input patterns to the various classifiers from a circuit simulator. Validation techniques, such as N-fold cross-validation and bootstrap methods, are employed for evaluating the robustness of network models. The trained networks are evaluated for their performance using test data on the basis of percent error rates obtained via cross validation and bootstrap techniques, time efficiency, memory size, generalization ability to unseen faults.

1. INTRODUCTION

Recently, neural network-based fault diagnosis and testability analysis of circuits and systems have become significant areas of research. This is primarily because neural networks excel at recognition and classification types of problems [1-6]. These techniques isolate faults by creating classifiers to identify faulty components or failure modes within components. The classifiers are trained using data obtained from Monte-Carlo simulations of the Unit under Test (UUT) under normal and various faulty scenarios. Moreover, the neural network models render themselves naturally to built-in self-test and on-line monitoring of the UUT.

The VTB used to generate fault-test dependency data is based on the Saber's TESTIFY tool (a fault simulation tool by Analogy Inc. [23-25]. TESTIFY provides information on the normal behavior of the UUT, and information on how well the set of tests specified by a test designer will detect the presence of faults in the system. The inputs to TESTIFY are: (1) a SPICE description of the UUT [25], (2) a set of test stimuli applied to the UUT, (3) a list of test-points where outputs are measured, (4) a set of test measurement features (such as voltage level, gain, phase, overshoot, rise time, settling time, etc), and (5) a pre-defined fault universe of failure modes for the components. TESTIFY parses the model and subcircuit information from the netlist and identifies the key model parameters. It, then, conducts a set of Monte-Carlo runs for all the pre-defined test stimuli. During these runs, circuit component parameters are varied randomly within the specified tolerance ranges. The fault simulation tool then creates a memory queue of all possible faults in the UUT. These faults are inserted, sequentially, in the original UUT's SPICE description and simulated against all tests. The concomitant fault signatures are then stored in a database for processing via the neural models.

The network models evaluated include Restricted Coulomb Energy (RCE) Neural Network, Learning Vector Quantization (LVQ), Decision Trees (C4.5), and Fuzzy Adaptive Resonance Theory (FuzzyArtmap). The N-fold cross-validation and bootstrap methods are used to evaluate the robustness of network models. The trained networks are evaluated for their performance using test data on the basis of percent error rates, time efficiency, memory size, and generalization ability to unseen faults. Studying and rating the performance of the neural network models for various classification types of problems is the primary focus of this research.

The rest of the paper is organized as follows. Section 2 briefly discusses the four learning algorithms. Section 3 provides an assessment of the four algorithms on a set of benchmark datasets from the UCI Repository of Machine Learning Databases and Domain Theories. Section 4 presents the results of applying the learning algorithms to three mixed-signal circuits. Section 5 concludes with a summary and future research directions.

2. MACHINE LEARNING ALGORITHMS

A. The RCE Neural Network

The RCE neural network [7],[8],[28] consists of three layers of "neuron cells" with a full set of connections between the first and second layers, and a partial set of connections between the second and third layers, as shown in Fig. 1.

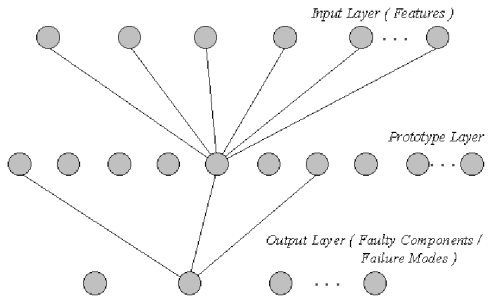


Figure 1. Three Layered RCE Network Structure

Each input layer cell represents a feature (a measurable characteristic) of an incoming pattern (an input signal) that the network assigns to some pattern class. The middle layer cells are called prototype cells, each of which contains information about an example of a learned pattern class that occurred in the training

[*] Vivek Kumar Rajan is at Intel 1 Drive, Portland, Oregon (email: rajan@intel.com)
Krishna Pattipati and Jie Luo are with the Department of Electrical and Systems Engineering, University of Connecticut, Storrs, CT 09269-3157, (email: Krishna@engr.uconn.edu)

data. Each cell on the output layer corresponds to a different pattern class represented in the training data set.

An RCE network cell is characterized by five elements: its class χ, its weight vector, ω, its cell threshold, λ, its pattern count, κ, and its smoothing factor, σ. During training, all but the smoothing factor play a role in prototype cell development. The prototype cell weight vector ω represents the set of weighted connections between the prototype cell and each of the input layer cells. In response to a signal on the input layer, each prototype cell computes a distance (Euclidean), d_i, between the input signal and the prototype vector stored in its weights via

$$d_i = \left[\sum_{j=1}^{N_D} \left(\omega_{ij} - x_j \right) \right]^{1/2} \qquad (1)$$

where

ω_{ij} = weight connecting i^{th} prototype cell and j^{th} input cell.

x_j = activity of j^{th} input cell (i.e., the j^{th} feature value of vector *x*)

N_D = number of input cells (i.e., dimension of feature space)

During training, a prototype cell becomes active, if the prototype-to-pattern distance *d* is less than the cell threshold, λ. This is called the activation rule and the prototype is said to fire. The network is trained through a sequence of input signals, each presented with its correct classification (a labeled training set). The supervised training procedure makes use of three mechanisms: prototype cell commitment, prototype threshold modification and prototype pattern count modification. The details are provided in [7,8,28].

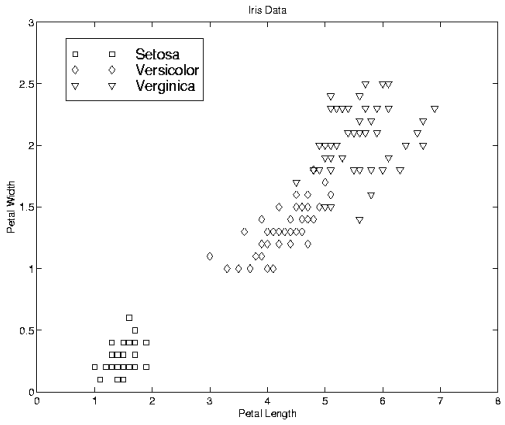
To illustrate the RCE classifier, we use the well-known Iris dataset. There are three varieties of Iris: Setosa, Versicolor and Verginica. The length and width of both petal and sepal were measured on 50 flowers of each variety. The original problem is to classify a new Iris flower into one of these three types based on four attributes (petal and sepal lengths and widths). To keep this example simple, we will look for a classification rule by which the varieties can be distinguished purely on the basis of two measurements: Petal Length and Width. We have 50 pairs of measurements (see Figure 2 (a)) of each variety from which to learn the classification. The results of applying the RCE network to the Iris dataset are shown in Fig. 2 (-b).

B. Learning Vector Quantization

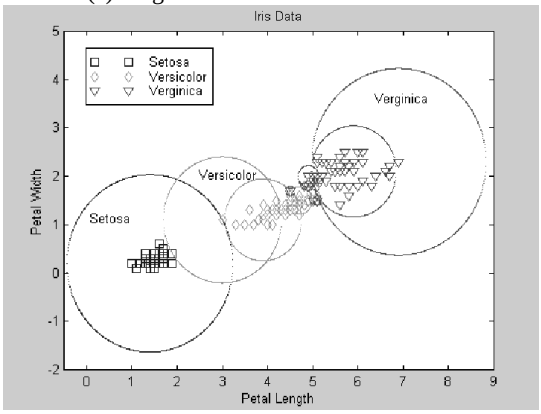
Learning vector quantization (LVQ) [9],[10] is a pattern classification method in which each output unit represents a particular class or category (several output units are used for each class.) The weight vector for an output unit is often referred to as a *codebook* vector for the class that the unit represents. During training, the output units are positioned (by adjusting their weights via supervised training) to approximate the decision surfaces of the classifier. The initial distribution of codebook vectors is also assumed to be known or is determined using traditional clustering algorithms (e.g., K-Means [11] or Class clustering [12]). After training, the LVQ net classifies an input vector by assigning it to the same class as the output unit whose weight vector (codebook vector) is closest to the input vector. The details of LVQ algorithm may be found in [9,10,28]. The results of applying the LVQ algorithm to the Iris dataset are shown in Fig. 3.

C. Decision Trees (C4.5)

A classification tree partitions the space of possible observations into sub-regions corresponding to the leaves and the label of the



(a) Original distribution of Iris dataset



(b) Classification result

Figure 2. Classification by RCE neural network: Iris dataset

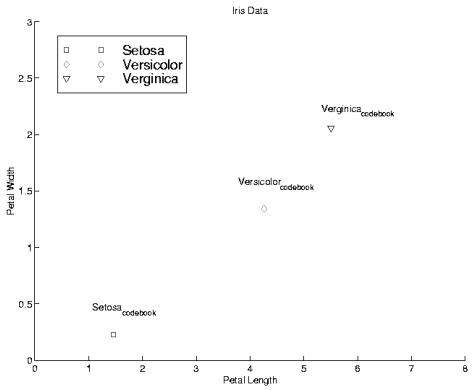


Figure 3. Classification by LVQ neural network: Iris dataset

leaf will classify each example. The decision tree structure generated is either:

- *a leaf*, indicating a class, or

- *a decision node* that specifies some test to be carried out on a single attribute value, with a branch (and the concomitant subtree) for each possible outcome of the test.

The process of constructing a decision tree from a set *T* of training cases is simple. Let the classes be denoted by $\{C_1, C_2, \cdots, C_k\}$. There are three possibilities:

- *T* contains one or more cases, all belonging to a single class C_j: The decision tree for *T* is a leaf identifying class C_j
- *T* contains no cases: The decision tree is again a leaf, but the class to be associated with the leaf must be determined from information other that *T*. For example, the leaf might be chosen in accordance with some background knowledge of the domain, such as the overall majority class. Usually, the most frequent class is selected.
- *T* contains cases that belong to a mixture of classes: In this situation, *T* is partitioned into subsets that are, or seem to be heading towards, single-class collection of cases. A test is chosen, based on a single attribute that has one or more mutually exclusive outcomes $\{O_1, O_2, \cdots, O_n\}$. *T* is partitioned into subsets $T_1, T_2, \cdots, T_n$, where T_i contains all the cases in *T* that have outcome O_i of the chosen test. The decision tree for *T* consists of a decision node identifying the test, and one branch for each possible outcome. The same tree-building mechanism is applied recursively to each subset of training cases, so that the *i*th branch leads to the decision tree constructed from the subset T_i of training cases

The algorithms for constructing decision trees are based on the concept of information gain and may be found in [27, 28]. The results of applying the C 4.5 algorithm to the Iris dataset are shown in Fig. 4.

D. Fuzzy Adaptive Resonance Theory (Fuzzy Artmap)

ARTMAP performs incremental supervised learning of recognition categories and multidimensional maps in response to input vectors presented in an arbitrary order. The fuzzy ARTMAP learns to classify inputs by a fuzzy set of features, or a pattern of fuzzy membership values between 0 and 1 indicating the extent to which each feature is present. This generalization is achieved by replacing the ART 1 modules [13],[14] of the binary ARTMAP system with fuzzy ART modules [15],[16]. Here, the ARTMAP system is trained several times on input sets with different orderings. Fuzzy ARTMAP has a small number of parameters and requires no problem-specific system crafting or choice of initial weight values. It learns each input as it is received on-line, rather than performing off-line optimization of a criterion function. The architecture of fuzzy ARTMAP is shown in Fig. 5. The details of fuzzy ARTMAP may be found in

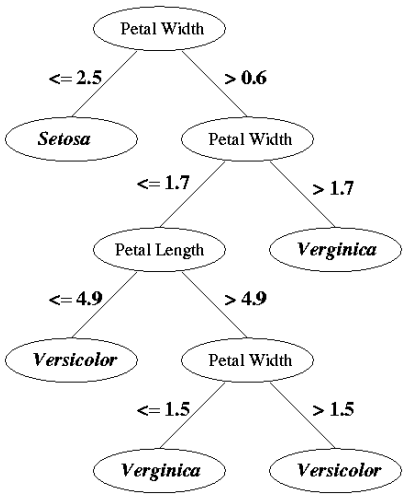


Figure 4. Classification by decision tree: Iris dataset

[15-17, 28]. The results of applying the fuzzy ARTMAP algorithm to the Iris dataset are shown in Fig. 6.

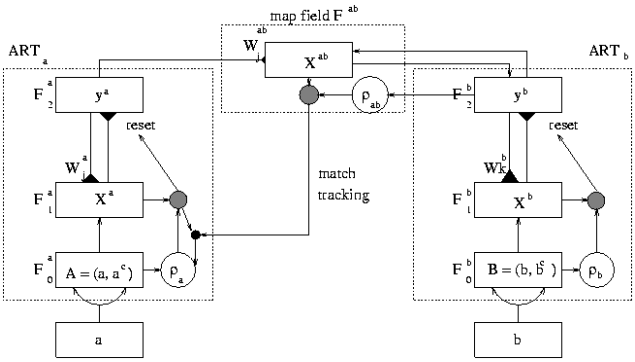


Figure 5. Fuzzy ARTMAP architecture. The ART_a complement coding preprocessor transforms the M_a vector *a* into the $2M_a$ vector $A = (a, a^c)$ at the ART_a field F_0^a. *A* is the input vector to the ART_a field F_1^a. Similarly, the input to F_1^b is the $2M_b$ vector (b, b^c). When a prediction by ART_a is disconfirmed at ART_b, inhibition of map field activation induces the match tracking process. Match tracking raises the ART_A search, which leads to activation of either an ART_a category that correctly predicts *b* or to a previously uncommitted ART_b category node.

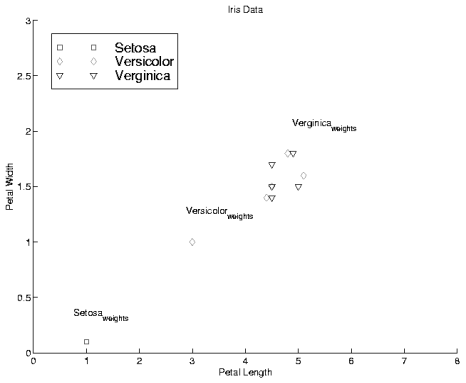


Figure 6. Classification by FuzzyArtmap neural network: Iris dataset

E. Methods of Comparison

In testing the accuracy of classification models, it is widely known that the error rates tend to be biased if they are estimates from the same set of data as that used to construct the network models [18]. The accepted procedures for predicting the error rates are described briefly in the following.

E.1 Cross-Validation

In its most elementary form, cross-validation divides data into *m* subsamples. Each sub-sample is predicted via a classification model trained with the remaining (*m*-1) sub-samples. The estimated error rate is the average error rate from these *m* subsamples. The leave-one-out method is an *m*-fold cross-validation with *m* set to the number of example datasets (*Q*).

E.2 Bootstrap

In statistical terms, the bootstrap is a non-parametric procedure for estimating parameters, and error-rates in particular. The basic idea is to re-use the original dataset (of size *Q*) to obtain new datasets (also of size *Q)* by sampling with replacement. Thus, to estimate the error rate in a small sample (of size *Q*), a large number *B* of bootstrap replicate samples are created, each sample being randomly chosen from the original sample. Sampling with replacement means that some data points may be omitted (on average about $e^{-1} = 36.8\%$ of data will not appear in the bootstrap sample). In addition, some data points will appear at least once in the bootstrap sample (on average $1-e^{-1} = 63.2\%$ chance). Each bootstrap sample is used to construct a classification model which is then used to predict the classes of those original data that were *unseen* in the training set. This gives one estimate of the error rate for each bootstrap sample. The average error rate of the original sample is then given by

$$e_{EFRON} = 0.368 \times e_{test-set} + 0.632 \times e_{training-set} \qquad (32)$$

where e_{EFRON} [18] is the average error rate of the original sample, $e_{test-set}$ is the error rate of the test-set and $e_{training-set}$ is the error rate of the bootstrap sample.

E.3 Cross-Validation and Bootstrap - Application to Iris Dataset

We use Iris dataset to evaluate the error rates using cross-validation and bootstrap methods. Table I shows the ten-fold cross-validation results for iris dataset. Average Error rate is shown in the second column of Table I for each neural network. Figure 7 shows the boxplot for the iris dataset with *B*=200 bootstrap samples.

RCE	0
LVQ	0.0778
DT	0.0333
FuzzyARTMAP	0.0222

Table I
10-FOLD CROSS-VALIDATION FOR IRIS DATASET

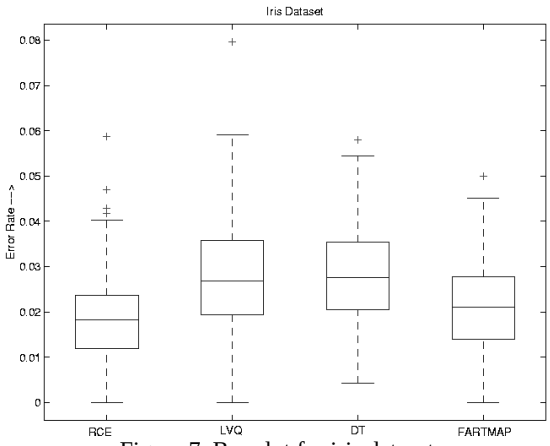


Figure 7. Boxplot for iris dataset

3. VALIDATION OF NEURAL MODELS VIA BENCHMARK DATASETS

A. Introduction

In the following, we consider several benchmark datasets downloaded from the archive at UCI Repository of Machine Learning Databases and Domain Theories. Some of these datasets have been described as being ``difficult categorization problems". The data sets are: (1) Wisconsin breast cancer (bcw), (2) Contraceptive method choice (cmc), (3) StatLog heart disease (hea), (4) Boston housing (bos), (5) BUPA liver disorders (bld), (6) StatLog vehicle silhoutte (veh), (7) Congressional voting records (vot), and (8) TA evaluation (tae). A summary of attribute features of the datasets is given in Table II.

Set	Size	No. of Classes	No. of Attributes	Noise Attributes
bcw	683	2	9	9 UI(1,10)
cmc	1473	3	9	6 N(0,1)
hea	270	2	13	7 N(0,1)
bos	506	3	13	12 N(0,1)
bld	345	2	6	9 N(0,1)
veh	846	4	18	12 N(0,1)
vot	435	2	16	14 C(3)
tae	151	3	5	5 N(0,1)

Table II:
CHARACTERISTICS OF THE DATASETS. THE LAST COLUMN GIVES THE TYPE OF ADDED NOISE ATTRIBUTES FOR EACH DATASET. THE NOTATION "*N*(0,1)" DENOTES THE STANDARD NORMAL DISTRIBUTION, "*UI*(*m*,*n*)" A UNIFORM DISTRIBUTION OVER THE INTEGERS *m* THROUGH *n* INCLUSIVE. THE ABBREVIATION *C*(*k*) STANDS FOR *UI*(1,*k*)

B. Experimental Setup

The attributes for each dataset are assumed to be continuous values for all neural models. In order to increase the number of datasets and to study the effect of noise attributes on each algorithm, we created eight new datasets by adding independent noise attributes. The numbers and type of noise attributes added are given in the right most column of Table II. The name of each new dataset is the same as the original dataset except for the addition of a **'+'** symbol. For example, the **bcw** dataset with

noise added is denoted by **bcw+**. For each dataset, we used ten-fold cross-validation scheme to estimate the error rates of an algorithm.

C. Results

The mean error rates of each algorithm over the datasets are shown in Table III. The minimum and maximum error rates are given for each dataset in the second last two columns of Table IV. Let p denote the smallest observed error rate in each row (i.e., dataset). If an algorithm has an error rate within one standard error of p, we consider it to be close to the best and indicate it by a * in Table III. The standard error rate is estimated as $\sqrt{\frac{p(1-p)}{Q}}$, where p is the error rate from a cross-validation estimate, and Q denotes the size of the training set. The bootstrap results show similar trends [28].

D. Analysis of Training Time and Memory Size

An analysis of training time and memory requirements of the four algorithms showed the following:

- Training time required by RCE neural network increases significantly with increases in the training set size. This can be explained as follows. Several training passes are required because a reduction in the size of a prototype's influence field during training

Dataset	RCE	LVQ	DT	Fuzzy-Artmap
bcw	0.04384*	0.04975*	0.04246*	0.05263
bcw+	0.07006	0.06005	0.04834*	0.04981*
cmc	0.42469*	0.54871	0.47185	0.5451
cmc+	0.59525	0.54595	0.48951*	0.58661
hea	0.3625	0.44444	0.2*	0.28519
hea+	0.39167	0.46667	0.21852*	0.41852
bos	0.32943	0.34777	0.2234*	0.28811
bos+	0.32521	0.34973	0.26704*	0.35478
bld	0.41917	0.43238	0.32905*	0.43333
bld+	0.44417	0.43238	0.32905*	0.43333
veh	0.3625	0.37733	0.27692*	0.32728
veh+	0.3626	0.37503	0.27216*	0.40186
vot	0.07226	0.09575	0.03409*	0.10058
vot+	0.20477	0.17756	0.05484*	0.11708
tae	0.38929*	0.41708*	0.4375	0.34458*
tae+	0.52238*	0.51708*	0.53583*	0.54917*

TABLE III
ERROR RATES FOR EACH DATASET FOR THE FOUR NEURAL MODELS

may result in its failure to identify patterns which, when initially presented, fell within its larger influence field. Eventually, however, a training pass will occur in which no new prototypes are committed and no prototypes have their influence fields reduced. This accounts for the relatively longer training times required by the RCE neural network for larger datasets;

- Training times for decision trees and fuzzy artmap network are similar;
- LVQ neural model requires the least training time for a given sample size. This is because no new codebook vectors are formed during the training phase; only their relative positions in the feature space are modified;
- Codebook vectors required to store a fully trained LVQ network does not depend on sample size, and remains constant with increase in sample size. This can be explained as follows. During training phase, LVQ network only modifies the relative positions of codebook vectors and does not add any new codebook vectors to the network. The initial set of codebook vectors fed to LVQ network is obtained from a clustering algorithm (K-Means or Class clustering);
- Number of nodes required to store a decision tree grows linearly with increases in sample size. For large training sets, a larger number of nodes is required to form a decision tree with high classification accuracy;
- Number of weights needed to store a fully trained fuzzyartmap network increases with increase in sample size. A large number of representative weights is required to construct a trained network for a larger training set.

Dataset	Min	Max	Std. Err
bcw	0.04246	0.05263	0.0077
bcw+	0.04834	0.7006	0.0082
cmc	0.42469	0.54871	0.0129
cmc+	0.48951	0.59525	0.013
hea	0.2	0.44444	0.0243
hea+	0.21852	0.46667	0.0251
bos	0.2234	0.34777	0.0185
bos+	0.26704	0.35478	0.0197
bld	0.30762	0.44619	0.0253
bld+	0.32905	0.44417	0.0253
veh	0.27692	0.37733	0.0154
veh+	0.27216	0.40186	0.0153
vot	0.03409	0.10058	0.0087
vot+	0.05484	0.20477	0.0109
tae	0.38929	0.4375	0.0397
tae+	0.51708	0.54917	0.0407

Table IV

MINIMUM, MAXIMUM AND STANDARD ERROR RATES FOR EACH DATASET, A'√' INDICATES THAT THE ALGORITHM HAS AN ERROR RATE WITHIN ONE STANDARD ERROR OF THE MINIMUM FOR THE DATASET.

4. APPLICATION TO CIRCUIT TESTING

A. Description of Circuits used for Validation

- **Brake**: This application is a brake-by-wire system, and includes a proportional solenoid valve and other

hydraulic and mechanical elements, as well as electrical and electronic devices for sensing and control (see Figure 8). The schematic of this circuit shows the simulation representation of an electronically controlled hydraulic braking system. The battery powered DC motor and hydraulic pump provide system pressure. A check valve, in series with a solenoid controlled two-way valve, establishes the controlled braking pressure at the midpoint. The solenoid valve is configured with pressure feedback on the spool, through a damping orifice. This inner feedback loop provides simple pressure regulation. The actual operating pressure is then adjustable with solenoid current, because the valve spool and the solenoid armature are rigidly connected. An outer pressure control loop uses an electronic sensor to measure the actual brake pressure. This measurement is compared with a commanded reference voltage

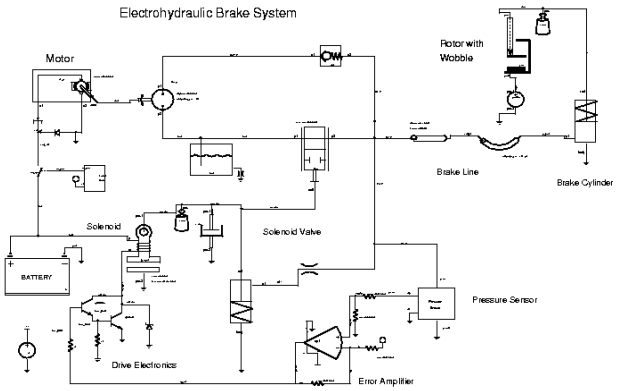


Figure 8. Schematic diagram of electrohydraulic brake system.

presumably coming from a brake pedal position sensor. The difference signal drives the base of a dual-transistor amplifier, which controls the solenoid current. The brake assembly is modeled as a single acting cylinder with spring return, plus attached load mass and a mechanical hard stop or travel limit. The hard stop models the contact point and compliance between the brake shoe and the rotor. A sinusoidally varying position source represents the instantaneous rotor transverse displacement. This allows simulation of the effects of rotor wobble on brake pressure regulation. Pressure is applied to the brake through a long rigid line, as well as a short flexible hose;

- **OscMos**: A pure analog hierarchical circuit that includes MOS level for the inverters, a "NAND" gate, several resistor-capacitor-diode (RCD) and RC circuits (see Figure 9). The proper loop delay characteristics are achieved using resistor-capacitor (RC) networks on the top level schematic. This schematic represents a relaxation oscillator which can be used as a clock to a analog-to-digital converter; and

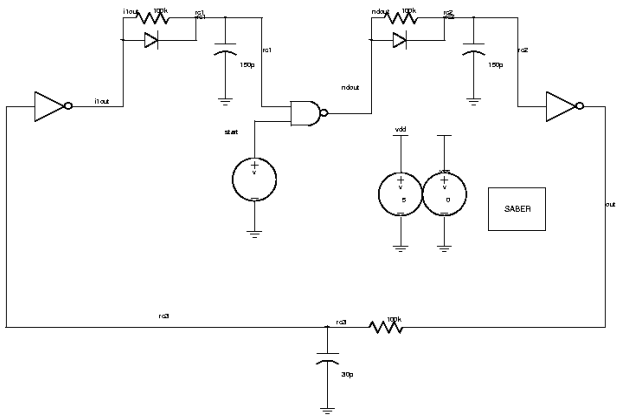


Figure 9. Schematic diagram of mos oscillator system.

- **AvgFwdConv**: This configuration contains the averaged model for the two switch forward converter and is used to design the feedback compensation (see Figure 10). This circuit is used to perform several simulations/analyses.

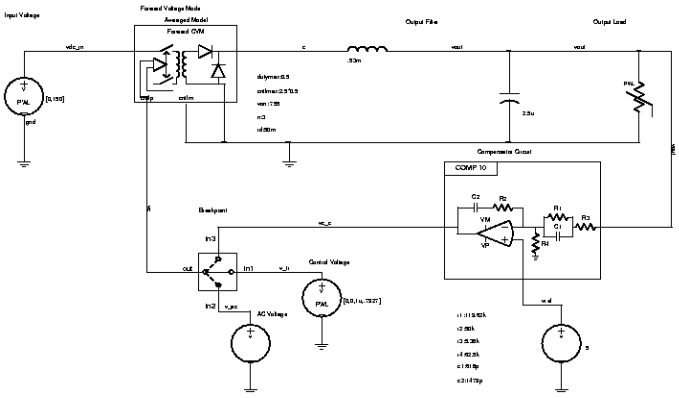


Figure 10. Schematic diagram of average forward converter system.

E. Mixed-Signal Fault Analysis

We present the mixed-signal fault analysis with the help of "Electrohydraulic Brake" system shown in Figure 8. The system is activated at 10msec and pressure builds to its maximum in approximately 200msec. At 400msec, the control voltage is reduced and the pressure falls to its operational level. After approximately 300msec of ``static" operation, the control voltage is increased, corresponding to activation of the brake pedal. System pressure rises accordingly. Figure 11 shows the transient analysis response of the circuit in the nominal mode. Figure 12 shows the same response with the diode shorted. The signals shown in Figures 11 and 12 are:

- Pump outlet pressure (*p_pump*),
- Brake pressure seen at the wheel cylinder (*p_brake*),
- Drive circuit input voltage (op_out), and
- Drive circuit output voltage (collector).

Nominal fault patterns for each circuit are obtained by performing 500 Monte-Carlo runs, and the fault patterns are obtained by performing 50 Monte-Carlo runs for each test. The

details of dataset obtained for each circuit are shown in Table V. Figure 13-15 shows the details of fault analysis performed using the TESTIFY tool for each circuit. The boxplot analyses for the above circuits with 200 bootstrap samples are shown in Figures 16-18.

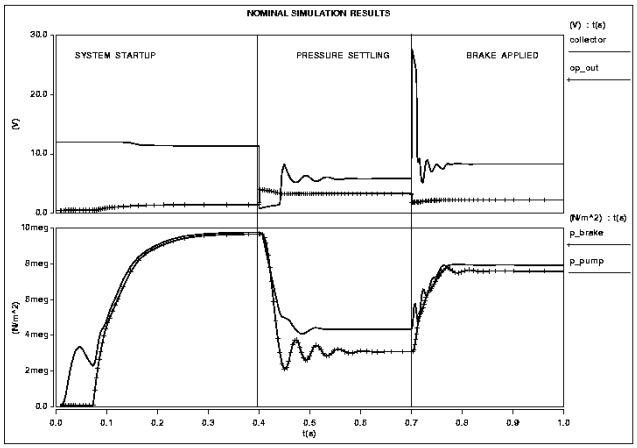


Figure 11. Nominal simulation plot for electrohydraulic brake system. Transient analysis is performed on the circuit during the first second in one micro second time steps.

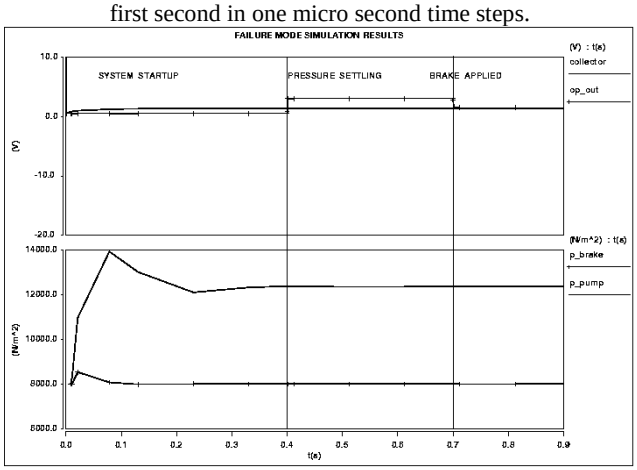


Figure 12. Failure mode simulation plot for electropydraulic brake system. The free wheeling diode across the motor is in short failure mode.

Circuit Name	Size of Dataset	No. of Classes	No. of Attributes
Brake System	1200	15	11
OscMos System	1450	20	9
AvgFwdConv System	1000	11	15

Table V
CHARACTERISTICS OF CIRCUIT DATASETS

Tests DC operating point analysis; and Transient analysis during the first second in one micro second time steps.

Measurements
- Drive circuit input voltage during system startup and pressure recovery (op_out);
- Brake pressure as seen at the wheel cylinder during system startup and pressure recovery (p_brake);
- Risetime of brake pressure as seen at the wheel cylinder (p_brake);
- Drive circuit output voltage during system startup and pressure recovery (collector);
- Pump outlet pressure during system startup and pressure recovery (p_pump);
- Solenoid armature position during system startup and pressure recovery (armature).

Faults

Diode	Open, Short;
Op-amp	Input open, Input short to V_{CC}, Output Open;
Transistor	Base open, Base short to emitter, Base open, collector open, emitter open;
Resistor	Open, Short.

Figure 13. Electrohydraulic brake system: analysis details

Tests DC operating point analysis; and Transient analysis for 100 micro seconds in 10 nano second time steps.

Measurements
- Oscillator output voltage (out);
- Nand gate input voltage (rc1);
- Inverter 2 input voltage (rc2);
- Buffer output voltage (rc3);
- Nand output voltage (ndout);
- Inverter 1 output voltage (i1out);
- Nand gate input voltage (start);
- Risetime of oscillator output (out); and
- Risetime of Buffer output (rc3).

Faults

Capacitor	Open, short;
Diode	Open, Short; and
Inverter (MOS)	Source open, Source short to ground, drain short to source, ground open, drain open, ground short to drain;
Nand (MOS)	Source open, Source short to ground, drain short to source, ground open, drain open, ground short to drain;

Figure 14. OscMos system: analysis details

Tests DC operating point; and Transient analysis for 500 micro seconds in 1 micro second time steps.

Measurements
- Peak-to-peak output voltage of compensation circuit (vout);
- Amplitude of output voltage of compensation circuit (vout);
- Minimum value of output voltage of compensation circuit (vout);
- Slewrate of output voltage of compensation circuit (vout);
- Slope of output voltage of compensation circuit (vout);
- Average input voltage to the breakpoint switch (v_tr);

	RMS value of output voltage of compensation circuit (vout);	
	Highpass 3dB point of output voltage of compensation circuit (vout);	
	Peak-to-peak control output voltage (vc_c);	
	Amplitude of control output voltage (vc_c);	
	Upper value of control output voltage (vc_c);	
	RMS value of control output voltage (vc_c);	
	Slewrate of Pulse-Width-Modulator (PWM) output (c);	
	Middle value of output voltage of compensation circuit (vout); and	
	Risetime of Pulse-Width-Modulator (PWM) output (c);	
Faults	Capacitor	Open, Short;
	Op-amp	Input open, Input short to V_{CC}, Input short to V_{EE}, Output Open; and
	Resistor	Open, Short

Figure 15. Averaged forward converter system: analysis details

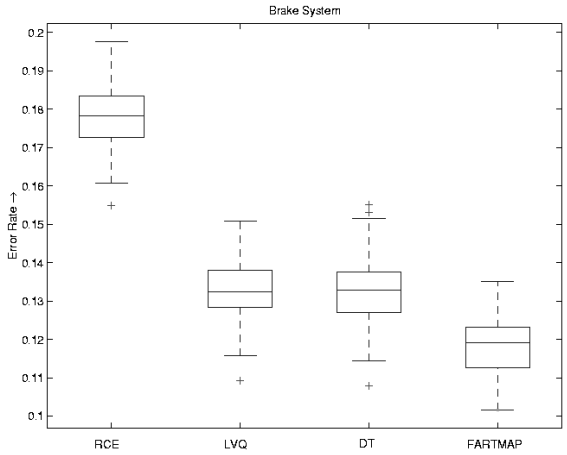


Figure 16. Boxplot diagram for electrohydraulic brake system.

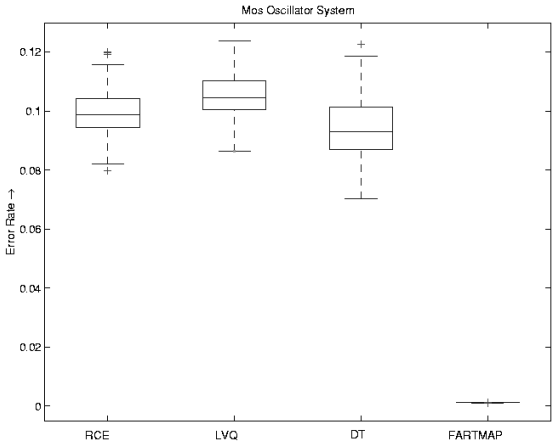


Figure 17. Boxplot diagram for mos oscillator system

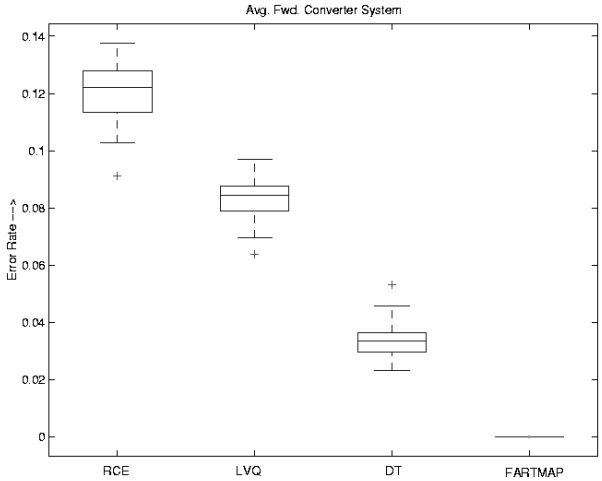


Figure 18. Boxplot diagram for forward converter system

Following conclusions can be drawn from Figures 16-18:

- The neural network models have consistent performance in identifying unseen faults provided the parameter variations are within certain limits (usually $\pm 20\%$). All the models are also consistent in identifying novel patterns which were unseen during the training phase. This establishes the generalization ability of the network models;
- FuzzyArtmap shows the best performance in identifying novel patterns during the test phase. The Fuzzyartmap network was trained with complement coding option being "on", which means that, both the on cell and off cell response for each feature were stored in the weight vectors of the network;
- Decision trees also show good performance in identifying novel patterns during the test phase;
- LVQ network has relatively higher error rates, which can be due to the lack of sufficient number of codebook vectors to represent each fault (class) in the feature space; and
- RCE network has the maximum error rates among all the four network models. This is because of the lack of sufficient number of prototypes for proper representation of each fault (class) in the feature space.

5. CONCLUSIONS

The neural network toolbox, even with its limited capabilities, provides an object-oriented simulation environment for neural network-based fault diagnosis. Our results show that the accuracy of diagnostic models is a strong function of the fidelity of the data extraction tool and the range of data in modeling various nominal and failure modes. The neural network models lend themselves naturally to on-line monitoring of systems, where measurement data from various sensors is continuously available.

With care, neural networks perform very well as measured by error rates. The observed differences in error rates can arise from:

- Suitability of the basic neural models for given datasets;

- Sophistication of default procedures for parameter settings; and
- Sophistication of the user in the selection of options and tuning of parameters.

A number of research issues need to be addressed to enhance the features of neural network toolbox. The enhancements include:

- Inclusion and rigorous testing of other neural networks (e.g., Multi-layer perceptrons, Radial basis functions, Probabilistic neural network, etc.);
- Parallel fault simulations, training and validation; and
- Committees of networks using the techniques of decision fusion [1,2, 29].

6. REFERENCES

[1] Bishop, C.M., *Neural Networks for Pattern Recognition*, Clarendon Press, Oxford, 1997.

[2] Haykin S., *Neural Networks: A Comprehensive Foundation*, New York: Macmillan College Publishing, 1994.

[3] Spina, R, Upadhyaya, S., "Linear Circuit Fault Diagnosis Using Neuromorphic Analyzers," *IEEE Transactions on Circuits and Systems-II: Analog and Digital Signal Processing*, Vol. 44, No. 3, March 1997, pp. 188-196.

[4] Spina, R, and, Upadhyaya, S., "Fault Diagnosis of Analog Circuits using Artificial Neural Networks as Signature Analyzers," *Proc Rochester Int. ASIC Conf*, Sept. 1992, pp. 357-362.

[5] Srinivasan, A, and, Batur, C., "Hopfield/ART-1 Neural Network-Based Fault Detection and Isolation," *IEEE Trans. on Neural Networks*, Vol 5, No 6, Nov. 1994.

[6] Yan, L, and Xiangying, W., "On Fault Diagnosis of Analog Circuits and Tolerance using Simulated Annealing Optimization Algorithm," *Proc 1992 International Conference on Industrial Electronics, Control, Instrumentation and Automation*, IEEE 1992.

[7] Reilly, D. L., "The RCE Neural Network," in J.D. Irwin (Ed.), *CRC Press Industrial Electronics Handbook*, pp. 1025-1037, 1997.

[8] Michael J. Hudak, "RCE Classifiers: Theory and Practice," *Cybernetics and Systems: An International Journal*, pp 483-515, Vol. 23, 1992.

[9] Kohonen, T., *Self-Organizing Maps*, Berlin: Springer. 1995.

[10] Kohonen, T., Jari Kangas, Jorma Laaksonen, Kari Torkkola, "LVQ_PAK: A program package for the correct application of Learning Vector Quantization algorithms," *Proc. of the Int. Joint Conf. on Neural Networks*, pp 725-730, Baltimore, June 1992. IEEE.

[11] MacQueen J., "Some methods for classification and analysis of multivariate data," *Proc. 5th Berkeley Symposium on Probability and Statistics*, University of California Press, Berkeley, 1967.

[12] Musavi, M. T., Ahmed, W., Chan, K. H., Faris, K. B. and Hummels, D. M., "On the Training of Radial Basis Function Classifiers," *Neural Networks*, Vol. 5, pp. 595-603, 1992.

[13] G. A. Carpenter, S. Grossberg, "A massively parallel architecture for a self-organizing neural pattern recognition machine," *Computer Vision, Graphics and Image Processing*, Vol. 37, pp. 54-115, 1987.

[14] G. A. Carpenter, S. Grossberg, *Pattern Recognition by Self-Organizing Neural Networks*, Cambridge, MA: MIT Press, 1991.

[15] G. A. Carpenter, S. Grossberg and D. B. Rosen, "Fuzzy ART: An Adaptive Resonance Algorithm for rapid, stable classification of analog patterns," *Proc. Int. Join Conf. Neural Networks*, Vol. II, pp. 411-420.

[16] G. A. Carpenter, S. Grossberg and D. B. Rosen, "Fuzzy ART: Fast Stable Learning and categorization of analog patterns by an adaptive resonance system," *Neural Networks*, Vol. 4, pp. 759-771, 1991.

[17] G. A. Carpenter, S. Grossberg, N. Markuzon, J. H. Reynolds and D. B. Rosen, "Fuzzy ARTMAP: A Neural Network Architecture for Incremental Supervised Learning of Analog Multidimensional Maps," *IEEE Trans. on Neural Networks*, Vol. 3, No. 5, Sept. 1992.

[18] J. David Irwin, *The Industrial Electronics Handbook*, CRC Press, 1996.

[19] Bradley Efron, *The Jackknife, the Bootstrap and Other Resampling Plans*, SIAM, 1982.

[20] C. J. Merz and P. M. Murphy, *UCI Repository of Machine Learning Databases*, Department of Information and Computer Science, University of California, Irvine, CA, 1996.

[21] Linda S. Milor, "A Tutorial Introduction to Research on Analog and Mixed-Signal Circuit Testing," *IEEE Transactions on Circuits and Systems-II: Analog and Digital Signal Processing*, Vol. 45, No. 10, Oct. 1998.

[22] J. W. Bandler and A. E. Salama, "Fault diagnosis of analog circuits," *Proc. IEEE*, Vol. 73, Aug 1985.

[23] Analogy Inc., "*Behavioral Fault Simulation of Large Mixed-Signal UUT's Using the Saber Simulator*," Technical Conference Paper, http://www.analogy.com/test/literature/ATC98Final.doc, 1998.

[24] Analogy Inc., "*Using Simulation to Improve Fault Coverage of Analog and Mixed-Signal Test Program Sets*," Tech. Conference Paper, http://www.analogy.com/test/literature/ATCpress97.doc, 1997.

[25] Analogy Inc., "*TESTIFY - Data Sheet*," Data Sheets, http://www.analogy.com/test/literature/TestifyDataSheet.pdf, 1999.

[26] Rashid, M. H., "*SPICE for Circuits and Electronics Using PSpice*," Second Edition, Printice Hall, Englewood Cliffs, NJ 07362, ISBN 0-13-124652-6.

[27] Quinlan, J.R., *C4.5: Programs for machine Learning*, : San Mateo: Morgan Kaufmann, 1993.

[28] Rajan, V., *A Neural Network Toolbox for Classification Problems*, MS Thesis, Dept. of ESE, Univ. of Connecticut, Storrs, CT, July 1999.

[29] Dasarathy, B. (Ed.), *Decision Fusion*, Los Alamitos, CA: IEEE Computer Society Press, 1994.