Training Neural Network for Machine Intelligence in Automatic Test Pattern Generator

Soham Roy, Spencer K. Millican, and Vishwani D. Agrawal
Department of Electrical and Computer Engineering
Auburn University, Auburn, AL 36849-5201
{*szr0075, millican, agrawvd*}*@auburn.edu*

***Abstract*—Recent research shows that an artificial neural network (ANN) can combine multiple heuristics to guide an automatic test pattern generator (ATPG) with fewer backtracks than required by guidance from any single heuristic. Thus motivated, we develop a new training method to include multiple heuristics. Our ANN has a single output neuron and a single layer of hidden neurons, which is sufficient to accommodate the training data volume. Conventional PODEM ATPG applied to hard-to-detect and easily detectable faults in selected benchmark circuits provide training data for nodes marked as "success" if the backtrace leads to a test or "failure" if it results in backtrack. ATPG data of a fault is used for training only if backtracks in the ANN-guided ATPG decrease. Circuit parameters added to training include input-output distances and testability values from COP (controllability and observability program) for signal nodes. Compared to the ANN guidance in previous studies, the proposed training method is found to require fewer total backtracks for all faults in any circuit from ISCAS'85 and ITC'99 benchmarks.**

***Index Terms*—Artificial neural network (ANN), ATPG, Digital testing, Machine learning**

I. Introduction

Automatic test pattern generation (ATPG) belongs in a set of NP-hard problems [1]. Such complexity may force one to try all possible circuit input vectors to find a test for a fault, but this is impractical for large circuits. Popular ATPG algorithms [2], [3] trace backward (i.e., "backtrace") from an interior node to a primary input (PI) to assign logic values. This action may or may not lead to a test, and in the latter case, the algorithm *backtracks*, or reverses, the PI assignment. To avoid backtracking, *heuristics* (i.e., rules based on a designer's intuition) select backtrace directions from available choices.

Choosing the best path during backtrace is a vital problem in ATPG. A recent study [4] showed that machine intelligence (MI) in the form of an artificial neural network (ANN) can replace any heuristic in PODEM [2] and speedup ATPG. However, some circuits did not exhibit any noticeable improvement in the backtracking performance. The ANN training was ad hoc and rudimentary, therefore, formulated training may elevate ANN-guided PODEM's performance. This study examines such a training method: the training recursively uses ATPG data generated from hard-to-detect as well as easily tested faults, resolves conflicts in data patterns (e.g., when different ANN outputs come from similar inputs), and discards data that does not improve the guidance of ANN. Computer-aided design (CAD) tool developers are unwilling to share the source code of commercial EDA software. Buying these EDA tools will only give us executables, making it impossible to run our experiments using them. Therefore, we ran our experiments using our own CAD programs. At present, ANN-guided ATPG is not typical practice, and this study will optimistically galvanize the CAD tool developers' minds to incorporate ANNs in their ATPG tools. The specific contributions of this study are as follows:

- A technique to resolve conflicts among ANN training data. Main benefit of conflict resolution is improved ANN accuracy as evident from reduction in the mean square error (MSE) during training.
- A technique for selecting faults using a recursive training method that dynamically trains an "evolving" ANN as opposed to training with a preselected set of faults. ATPG data from a fault is retained only if the trained ANN guidance continues to further reduce backtracks over those from conventional heuristics (distance and COP). Both hard-to-detect and easy-to-test faults are sampled as opposed to only the former [4]. This prevents the training data from being overwhelmingly "failure" oriented.
- An improved assessment of the quality of the ANN guidance by modifying the backtrack count procedure. We count backtracks only for faults that are detected or found redundant; earlier work [4] also included backtracks from aborted faults, which distorts the count.

The remainder of this article is organized as follows. Section II briefly describes prior work on ATPG including MI applications. Section III outlines the motivation leading to the present work and summarizes the objectives. Section IV describes the proposed training techniques. Section V provides experimental exploration of the proposed training technique using benchmark circuits. Sections VI and VII highlight some future directions and conclude the article, respectively.

II. Prior Work

In early work, D-algorithm [5] provided a comprehensive search method over the space of 2^N states for N signal nodes in the circuit. An improvement came in the form of PODEM [2], where search space is reduced to $2^{\#PI}$ vectors for $\#PI$ PIs. Due to its simplicity, PODEM became a popular ATPG algorithm, and it could easily use heuristics like distance [2], COP [6], SCOAP [7], CAMELOT [8], and others to guide its backtracing decisions. FAN [9] further modified

Proceedings of 34th International Conference on VLSI Design (VLSID 2021), pages 1-6
February 20-24, 2021 (Virtual Event)

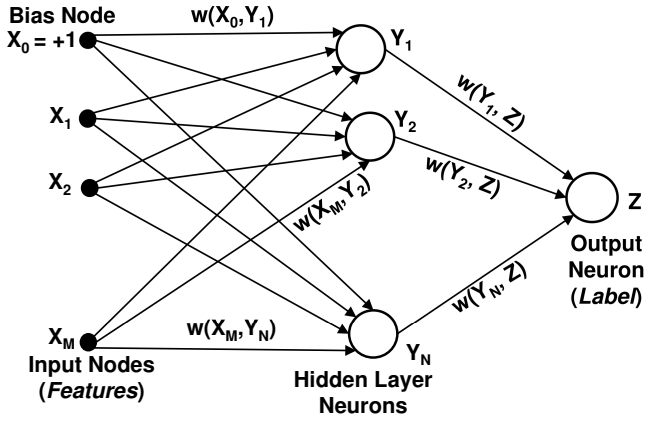

Fig. 1. The ANN used in [4] consists of inputs (bias X_0 fixed at 1.0, and M inputs, X_i), a hidden layer of N neurons, $\{Y_j\}$, and a single output neuron, Z. Directed edges connect input to all hidden layer neurons, and hidden layer neuron to the output neuron.

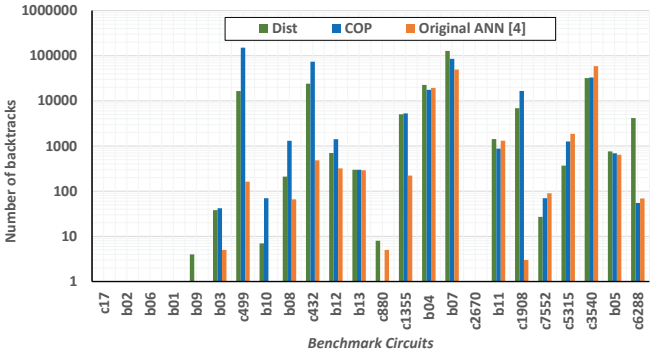

Fig. 2. Previous results [4]: Backtracks in PODEM ATPG for 100 hard-to-detect checkpoint stuck-at faults (including detected and redundant). The original ANN guidance (orange bars) reduced backtracks over conventional heuristics (green-distance, blue-COP).

the search space from all PIs to headlines: it used PODEM-like procedures with heuristic guidance taken from fanout weights. This work uses PODEM, which readily adapts to any heuristic, including those based on MI [4].

Applications of MI or ANN in VLSI design and test started in the late 90s and gained popularity in the last two decades [10]–[12]. ATPG using ANN was first attempted by Chakradhar et al. [13] by modeling digital circuits as an ANN: stable states of an ANN model (i.e., minimum energy states) provided tests. Although effective, such ATPG is time-consuming and complicated because the ANN energy profile contains local minima traps. Other testing problems like scan chain diagnosis [14], identification of fault models [15], testability measures [16], analog and RF circuit testing [17], [18], and test-point insertion (TPI) [19]–[22] have been solved using ANNs.

A recent study [4] applied MI to improve the efficiency of ATPG. The authors reported that an ANN could combine multiple heuristics to guide ATPG. Their ANN consists of an input layer, a single hidden neuron layer, and an output neuron, as illustrated in Fig. 1, and the output of a neuron lies in the [0,1] range. The values of inputs, referred to as features, $X_0, X_1, \cdots, X_M$, were normalized to [0,1] to facilitate the ANN training. X_0, fixed at 1.0, is a bias input. Hidden neurons $Y_1, Y_2, \cdots, Y_N$ and a output neuron Z, called the "label", evaluate their outputs using respective inputs. The output value of a neuron is denoted as x_i, y_i or z. The directed edge from any neuron A to another neuron B, denoted by $w(A, B)$, carries a signed floating point value. The output of any neuron Y_j is calculated as,

$$y_j = f(\sum_{i=1}^{M} x_i \times w(X_i, Y_j)) \quad (1)$$

where f is the activation function [23] for which sigmoid function [24] was used.

$$f(v) = 1 - \frac{1}{1 + e^{-v}} \quad (2)$$

A training data pattern consists of inputs (features) and expected output (label) values. During training, for any given input features, the output label is computed and compared to the expected value of output. The square of difference between computed value and expected value of output neuron averaged over all training patterns is the mean squared error (MSE). Weights are adjusted in successive training "epochs" to minimize this MSE. This error can be further minimized by tuning hyper parameters like the number of hidden layers, number of hidden neurons per hidden layers, learning rates, activation functions, etc. Adam [25] is a typical optimizer prominently used for feed-forward ANN structures. The input features of the ANN contain the following: 1) The one-hot encoded gate-type of a gate driven by the circuit line being traced during ATPG; 2) from COP [6], CC (probability of setting a line to "1"), and CO (probability of line being observed at a circuit output); 3) the minimum distance between the line being traced and circuit PIs.

Improvement in ATPG performance was reported [4] for many benchmark circuits, but several circuits, including some large benchmarks (c3540, c7552, c2670, and c432), did not perform well. We will discuss these results in the next section, since they serve as the motivation for this work.

III. Motivation

To motivate this work, we repeat previous work [4] with minor changes. Training patterns were generated using PODEM applied to 100 hard-to-detect faults in the three highest depth circuits: c6288, c3540, and b05. The trained ANN was used to guide a PODEM applied to the 100 hardest-to-detect faults in each of the 24 benchmark circuits used in [4]. Figure 2 compares the total backtracks with those from conventional PODEM using distance [2] and COP [6] heuristics. One difference in our evaluation procedure, as stated in Section I, is that we did not include backtracks from aborted faults. We refer to this ANN as the "original ANN" to distinguish it from the ANN developed in this paper, identified as the "new ANN".

We make several observations from Figure 2. Circuit size increases from left to right and the first four small circuits and c2670 have no backtrack by any method. Also, for circuits b09, b10 and c2670, the ANN required no backtracks. Although ANN guidance satisfies the expectation of fewer backtracks than both conventional heuristics for several circuits, there are exceptions, especially among larger benchmarks. Notable among these are c7552, c5315 and c3540. For example, c3540 shows 32,008 (distance), 32,978 (COP), and 58,852 (ANN) backtracks used, similar observations were made in [4].

We believe remedying these outliers requires addressing several aspects of the ANN training procedure. Our investigation led us into following observations:

1) **Selection of training circuits.** In the previous work [4], the ANN was trained with ATPG data from high depth circuits: c6288, c3540 and b05. Since c3540 causes worry, we retrained the ANN with just c3540, but still the ATPG performance of this circuit did not improve. Therefore, we retained the three high depth circuits for training.
2) **Resolution of conflicts among training data patterns**. These data patterns may lead to increase in training error of the ANN and thus degrade its guiding ability. This is explained in Section IV-A.
3) **Selection of faults**. All faults do not provide useful training information, and preselection of faults for training may unnecessarily increase the training data volume without benefit. Section IV-B gives a novel method of training the ANN considering three aspects, i.e., the training error of ANN must be kept low by adjusting the number of hidden neurons, training data from a fault must be accepted only if guidance from the resulting ANN reduces backtracks, and the faults used for training data generation should not restricted to be hard-to-detect faults. In fact, a fault provides useful training data as long as it reduces backtracks.
4) **Forgetfulness of the ANN**. The ANN may enter a zone called "catastrophic forgetting" [26] when it forgets information contained in a large training data volume. Such consideration in training improvement is worth exploring in the future.

In this study, we discuss a new training strategies listed addressing the above (except Item 4) to improve the backtracking performance of ANN-guided PODEM. In previously reported work [4], the number of hidden neurons in the ANN was preselected, but a static ANN is incapable of absorbing increasing amounts of training data. The new training technique progressively adds training data after resolving conflicts in data patterns to further reduce backtracks while discarding data that does not accomplish this objective, and the ANN evolves through addition of hidden neurons to further reduce MSE during training.

Input Features												Output label
Gate type									COP(CC)	COP(CO)	Distance	Success/Failure
x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}	x_{12}	z
0	0	0	0	0	0	0	0	1	0.76	0.84	0.12	1
0	0	0	0	0	0	0	0	1	0.76	0.84	0.12	1
0	0	0	0	0	0	0	0	1	0.76	0.84	0.12	0
0	0	0	0	0	0	0	1	0	0.95	0.95	0.95	1
0	0	0	0	0	0	0	1	0	0.95	0.95	0.95	1
0	0	0	0	0	0	1	0	0	0.5	0.5	0.5	0
0	0	0	0	0	0	1	0	0	0.5	0.5	0.5	0
0	0	0	0	1	0	0	0	0	0.37	0.63	0.25	1
0	0	0	0	0	1	0	0	0	0.36	0.43	0.75	0
0	0	0	0	0	0	0	0	1	0.63	0.34	0.8	1

Fig. 3. An example of ANN training data patterns with conflicts. Note the first three patterns with identical inputs (features) and conflicting outputs (labels).

Input Features												Output label
Gate type									COP(CC)	COP(CO)	Distance	Success/Failure
x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}	x_{12}	z
0	0	0	0	0	0	0	0	1	0.76	0.84	0.12	0.67
0	0	0	0	0	0	0	1	0	0.95	0.95	0.95	1
0	0	0	0	0	0	0	1	0	0.95	0.95	0.95	1
0	0	0	0	0	0	1	0	0	0.5	0.5	0.5	0
0	0	0	0	0	0	1	0	0	0.5	0.5	0.5	0
0	0	0	0	1	0	0	0	0	0.37	0.63	0.25	1
0	0	0	0	0	1	0	0	0	0.36	0.43	0.75	0
0	0	0	0	0	0	0	0	1	0.63	0.34	0.8	1

Fig. 4. Example ANN training data patterns after resolving conflicts. The first pattern here replaces the first three patterns of Figure 4.

IV. Proposed Training Methodology

A. Resolving conflicts in training data

The ith training data pattern consists of an input vector $\{x_i\}$ of features and an output label z_i. The label and all features range in [0,1]. Two patterns, i and j, form a conflicting pair if $\{x_i\} \equiv \{x_j\}$ and $z_i \neq z_i$. This is because the ANN cannot be trained to produce different outputs for the same input. The presence of conflicting patterns in training data influences the training as indicated by non-decreasing MSE during training.

To remedy this, we collect patterns with identical features into groups and then replace each group by a single *representative pattern* with common features. Since these patterns are derived from actual ATPG runs, the label of a pattern is either 1 (indicating success) if it belongs in a backtrace leading to a test or 0 (indicating failure) if it results in a backtrack. We count the number of 1 and 0 labels in the group, and the representative pattern is given the label $\frac{count(1)}{count(0)+count(1)}$.

A conflict is illustrated by the sample training patterns in Figure 3, where the first three patterns have a conflict. This is resolved in Figure 4, where the three patterns are replaced by a single representative pattern with a label $\frac{2}{2+1} = 0.67$.

We collected training data from conventional heuristic-based PODEM applied to the training circuits (c6288, c3540 and b05), resolved all conflicts among training data patterns, trained the ANN, and integrated the ANN guidance into PODEM. We observed that pre- and post-conflict resolution MSE were 3% and 1%, respectively. Additionally, training

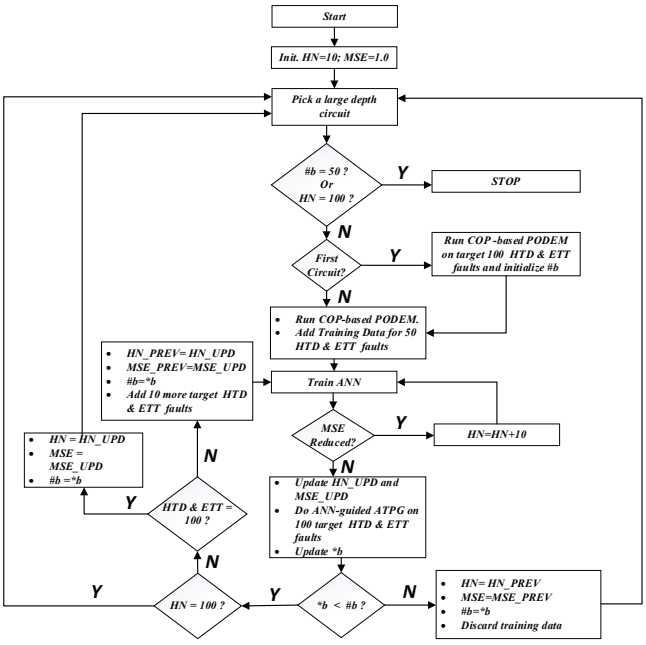

Fig. 5. Flow chart of proposed training methodology, including sub-procedures, recursive training by conventional heuristic-based PODEM, followed by "evolving" ANN.

patterns were compacted in this manner allowed more faults to be included during training.

B. Recursive training and evolving ANN

In previously reported work [4], PODEM with random heuristic generated the ANN training data. This can be time-consuming, difficult, and non-repeatable. Experimental results of Figure 2 show that COP [6] heuristic can provide reasonable ATPG performance (i.e., fewer backtracks or in the same ballpark regime) for the training circuits. In a recent study [4], PODEM with random heuristic was used to train ANN. That approach could be time-consuming and difficult to repeat. Therefore, PODEM guided by COP is used for training data generation.

ANN training quality largely depends on its structure and complexity. We examined the interrelation between the number of hidden neurons and MSE of the ANN to find a "sweet-spot" that guarantees efficient ANN training. This "sweet-spot" is found during the run-time of ANN's complexity determination step by either adding hidden neurons or restoring the previous optimal hidden neurons with corresponding training data, making the ANN effective with minimal MSE. Training data from PODEM using COP is obtained from a small set of faults and applied recursively to train ANN. This training continues to minimize MSE by adding small batches of faults to the training as long as they continue to improve the ANN quality as discussed later in Section IV-C. This makes our ANN dynamic in terms of hidden neurons.

C. Algorithm

The algorithm to develop an "evolving" ANN-guided ATPG is illustrated in Figure 5. We select large depth circuits, c6288, b05, and c3540, from ISCAS'85 and ITC'99 benchmarks [27], [28]. The ANN structure is same as in the previous study (Figure 1) and parameters are initialized as "hidden neurons (HN) = 10" and "MSE = 1.0". COP-based PODEM is applied to a set of 100 hard-to-detect faults, and number of backtracks is recorded "#b" to be minimized through recursive training. The training starts by using 50 hard-to-detect faults to generate a training database, followed by ANN training to check MSE and record the corresponding number of hidden neurons, simultaneously. This process continues until MSE starts saturating, and the corresponding MSE "MSE_UPD" and the number of hidden neurons "HN_UPD" are recorded and the ANN is re-trained. The similar process is continued for 100 easy-to-test faults. To evaluate the training, the ANN-guided ATPG is applied to 100 hard-to-detect faults of the same training circuit. If backtrack count decreases, then 10 more hard-to-detect faults are added to the training of the ANN, else the hidden neurons "HN_PREV" and MSE "MSE_PREV" are restored, discarding the corresponding training patterns from training database, and another circuit is selected to re-iterate the same process until one of the following conditions is satisfied: 1) the number of backtracks is reduced to 50; 2) 100 hard-to-detect faults are used for training; 3) the ANN contains 100 hidden neurons.

V. Experimental Results

All experiments were conducted on an industry-oriented workstation (Intel-i8700 microprocessors, 8GB RAM). Our design-for-test (DFT) tools are written in C++ and compiled using the MSVC++14.15 compiler with maximum optimization settings. PODEM is programmed for easy-to-plugin heuristics, distance [2], COP [6], the original ANN [4] or the new ANN, can be applied across 24 ISCAS'85 [27] and ITC'99 [28] benchmark circuits with no other modifications to the ATPG algorithm.

Figure 6 shows how this article's the new ANN performs on 100 hard-to-detect faults across benchmarks as compared to the original ANN [4]. In terms of backtracks, with few exceptions, the new ANN does the same or better in reducing backtracks compared to the original ANN [4], especially for larger circuits.

Our next experiment used all (testable and redundant) faults to show the efficacy of guidance by the new ANN using the proposed training technique. Table I shows the computation time of ATPG "CPU Time (ms)", "Backtrace count", and "Backtrack count". Clearly, the new ANN performs better (with fewer backtracks) than the original ANN [4], reaffirming the value of the proposed training technique of this article. Backtrack counts for all faults are shown in Figure 7.

Table I has some notable observations. First, c17, b02, and b01 have no reconvergent fanouts and therefore have no backtracks. Of course, there is no scope for reducing backtracks by new ANN guidance. We observe that the number of backtraces is either constant or reduced in these reconvergent fanout-free circuits by the new ANN over the original ANN [4]. Second, except c1908, c432, c499, b12, and b05, the new

TABLE I
PERFORMANCE OF PODEM ATPG FOR ALL CHECKPOINT FAULTS IN BENCHMARK CIRCUITS, GUIDED BY ORIGINAL ANN [4] AND NEW ANN (THIS PAPER). BOLDFACE NUMBERS SHOW REDUCED BACKTRACKS BY NEW ANN.

Circuit name	Original ANN [4]			New ANN (this paper)		
	CPU time (ms)	Backtrace count	Backtrack count	CPU time (ms)	Backtrace count	Backtrack count
c17	7	64	0	7	64	**0**
b02	41	236	0	41	236	**0**
b06	128	514	1	130	506	**0**
b01	121	514	0	117	498	**0**
b09	984	3293	23	1250	3239	**4**
b03	662	2166	78	605	1901	**8**
c499	7097	22418	933	6828	22888	965
b10	1784	3718	361	1448	3225	**181**
b08	1683	4420	804	1665	4205	**481**
c432	12610	26253	19840	13626	30150	21441
b12	24877	33814	7161	24496	33945	7482
b13	1720	5481	1063	1453	4871	**570**
c880	4014	11889	7	3481	10459	**0**
c1355	40231	58658	1498	35825	57788	**934**
b04	67329	66182	46423	43855	47139	**27110**
b07	14660	18851	10810	12741	16420	**7476**
c2670	48296	72347	29924	38816	65159	**18355**
b11	17397	21008	5673	14601	18242	**3641**
c1908	30616	39150	549	27685	35982	779
c7552	291393	297572	57874	214372	222395	**11183**
c5315	85702	105072	21782	69560	86429	**7321**
c3540	126861	82609	41842	100468	71492	**30529**
b05	56152	43078	21260	48830	39892	21540
c6288	457206	212968	29982	390805	180588	**16525**

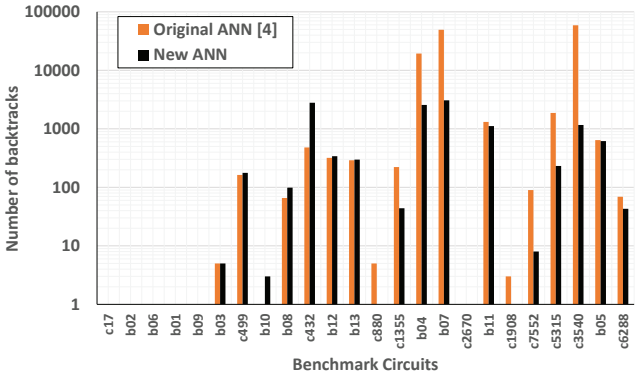

Fig. 6. New result: Backtracks in PODEM ATPG for 100 hard-to-detect checkpoint stuck-at faults (including detected and redundant). Formally trained ANN guidance (black bars) show reduced backtracks over the previous ANN [4] (orange bars).

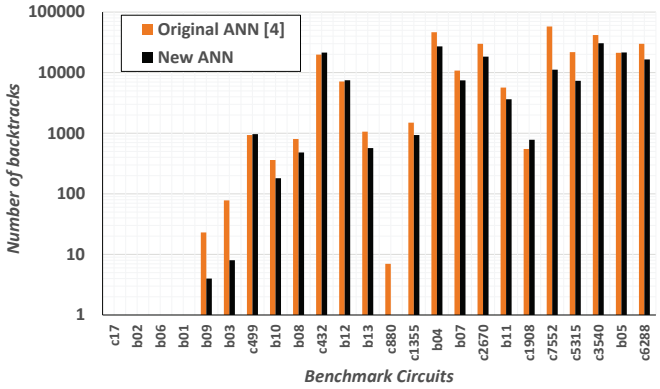

Fig. 7. New result: Backtracks in PODEM ATPG for all checkpoint stuck-at faults (including detected and redundant). Formally trained ANN guidance (black bars) show reduced backtracks over the previous ANN [4] (orange bars).

ANN exceeds expectations in terms of performance based on reductions in backtracks. Third, the new ANN is able to reduce backtracks and backtraces for b09 but the CPU time increases. Possibly more time per backtrace is used in order to reduce backtracks by expensive evaluation of weights and biases of ANN edges that involves matrix multiplication and computation of sigmoid [24] function. Forth, the new ANN heuristic does not perform as well on a few circuits compared to the original ANN [4], but, it still outperformed the conventional heuristics (like Distance [2] and COP [6]), except in case of c432. Fifth, Figure 8 illustrates that CPU time remains constant for smaller circuits, and a reduction for high depth circuits is observed perhaps due to simultaneous reduction in backtracks and backtraces.

VI. DISCUSSION AND FUTURE DIRECTIONS

This study provides several avenues to be explored. First, our focus was on reducing backtracks, but the performance of backtraces, particularly in reconvergent fanout-free circuits, can also be improved. Second, our exploration for eliminating backtracks has a cost in CPU time. Thus, a "sweet-spot" may be found where the reduction of backtracks would be optimum for minimizing CPU time. Third, finding untestable/redundant faults earlier can make ATPG for the entire circuit faster.

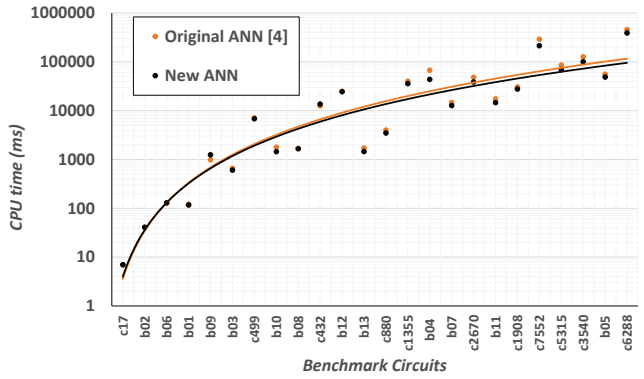

Fig. 8. New result: CPU time (ms) of PODEM ATPG for all checkpoint stuck-at faults (including detected and redundant). Formally trained ANN guidance (black line) shows some reduction in CPU time over the previous ANN [4] (orange line), especially for larger benchmarks.

Fourth, recent work [20] has demonstrated that arbitrary random circuits can generate limitless training data. Fifth, this study is demonstrated on academic benchmark circuits and not on larger industry standard circuits. The authors believe that the ANN-guided ATPG performance trends are quite promising (which is important) and likely to scale on a wider variety and larger set of designs to show broader capabilities in future. Finally, there may be other features, such as SCOAP [7] and those related to fanouts or reconverging signals, that can add to the capability of the ANN. Yet another area is to examine ANN structures beyond the single hidden layer [29].

We found that training with just hard-to-detect faults was not sufficient for obtaining a more useful ANN. Therefore, we had to include some easy-to-test faults in the training process. A possible reason is that the hard-to-detect faults may cover only some parts of the circuit topology while ignoring others.

VII. Conclusion

The methodical training of an ANN for guiding ATPG as presented here has benefits. Although many cases show significant improvements, there are circuits that demand more. Finding the most suitable training circuits remains an open problem. ANN training is only a one-time cost, after which the MI imparted to the ATPG can have long-term benefits. However, there is an added CPU time cost in every ATPG run because of the more complex computation required during the ANN-based decision. In an alternative implementation, some of those computations may be avoided by using lookup tables. Also, practical ATPG systems may combine a simple program (e.g., random vector ATPG) for easy faults and a complex program (e.g., ANN-based ATPG) for hard-to-detect faults.

References

[1] O. H. Ibarra and S. K. Sahni, "Polynomially Complete Fault Detection Problems," *IEEE Trans. Comp.*, vol. C-24, pp. 242–249, 1975.

[2] P. Goel, "An Implicit Enumeration Algorithm to Generate Tests for Combinational Logic Circuits," *IEEE Trans. Comp.*, vol. C-30, pp. 215–222, 1981.

[3] M. L. Bushnell and V. D. Agrawal, *Essentials of Electronic Testing for Digital, Memory and Mixed-Signal VLSI Circuits*. Springer US, 2013.

[4] S. Roy, S. K. Millican, and V. D. Agrawal, "Machine Intelligence for Efficient Test Pattern Generation," in *Proceedings of the IEEE Int. Test Conf.*, Nov 2020, pp. 1–6.

[5] J. P. Roth, W. G. Bouricius, and P. R. Schneider, "Programmed Algorithms to Compute Tests to Detect and Distinguish Between Failures in Logic Circuits," *IEEE Trans. Electronic Comp.*, vol. EC-16, pp. 567–580, 1967.

[6] F. Brglez, "On Testability Analysis of Combinational Circuits," in *Proc. Int. Symp. Circ. and Sys.*, Montreal, Quebec, Canada, 1984, pp. 221–225.

[7] L. Goldstein, "Controllability/observability analysis of digital circuits," *IEEE Trans. Circ. and Sys.*, pp. 685–693, 1979.

[8] R. G. Bennetts, C. M. Maunder, and G. D. Robinson, "COMELOT: a Computer-aided Measure for Logic Testability," *IEE Proceedings E - Comp. and Digital Techniques*, vol. 128, no. 5, pp. 177–189, 1981.

[9] Fujiwara and Shimono, "On the Acceleration of Test Generation Algorithms," *IEEE Trans. Comp.*, vol. C-32, pp. 1137–1144, 1983.

[10] M. Pradhan and B. B. Bhattacharya, "A Survey of Digital Circuit Testing in the Light of Machine Learning," *WIREs Data Mining Knowl. Discov.*, pp. 1–18, 2020.

[11] H. Stratigopoulos, "Machine Learning Applications in IC Testing," in *Proc. IEEE 23rd European Test Symposium (ETS)*, Bremen, Germany, Jun 2018, pp. 1–10.

[12] S. Roy, S. K. Millican, and V. D. Agrawal, "Special Session – Machine Learning in Test: A Survey of Analog, Digital, Memory, and RF Integrated Circuits," in *Proc. IEEE VLSI Test Symposium (VTS'21)*, 2021, pp. 1–10.

[13] S. T. Chakradhar, V. D. Agrawal, and M. L. Bushnell, *Neural Models and Algorithms for Digital Testing*. Springer, 1991.

[14] M. Chern, S.-W. Lee, S.-Y. Huang, Y. Huang, G. Veda, K.-H. H. Tsai, and W.-T. Cheng, "Improving Scan Chain Diagnostic Accuracy Using Multi-Stage Artificial Neural Networks," in *Proceedings of the 24th Asia and South Pacific Design Automation Conf. (ASP-DAC)*, Tokyo, Japan, Jan 2019, pp. 341–346.

[15] L. R. Gómez and H.-J. Wunderlich, "A Neural-Network-Based Fault Classifier," in *Proc. IEEE 25th Asian Test Symp. (ATS)*, Hiroshima, Japan, Nov 2016, pp. 144–149.

[16] Y. Ma, H. Ren, B. Khailany, H. Sikka, L. Luo, K. Natarajan, and B. Yu, "High Performance Graph Convolutional Networks with Applications in Testability Analysis," in *Proc. 56th ACM/IEEE Design Automation Conf. (DAC)*, Las Vegas, NV, USA, June 2019, pp. 1–6.

[17] H.-G. Stratigopoulos and Y. Makris, "Error Moderation in Low-Cost Machine-Learning-Based Analog/RF Testing," *IEEE Trans. Computer-Aided Design of Integrated Circ. and Sys.*, vol. 27, no. 2, pp. 339–351, Feb. 2008.

[18] D. Maliuk, H.-G. Stratigopoulos, H. Huang, and Y. Makris, "Analog Neural Network Design for RF Built-In Self-Test," in *Proc. Int. Test Conf. (ITC)*, Austin, TX, USA, Nov 2010, pp. 23.2.1–23.2.10.

[19] Y. Sun and S. K. Millican, "Test Point Insertion Using Artificial Neural Networks," in *Proc. IEEE Comp. Society Annual Symp. on VLSI (ISVLSI)*, Miami, FL, USA, Jul 2019, pp. 253–258.

[20] S. K. Millican, Y. Sun, S. Roy, and V. D. Agrawal, "Applying Neural Networks to Delay Fault Testing: Test Point Insertion and Random Circuit Training," in *Proc. IEEE 28th Asian Test Symp. (ATS)*, Kolkata, India, Dec 2019, pp. 13–18.

[21] S. Roy, B. Stiene, S. K. Millican, and V. D. Agrawal, "Improved Random Pattern Delay Fault Coverage Using Inversion Test Points," in *2019 IEEE 28th North Atlantic Test Workshop (NATW)*, Burlington, VT, USA, May 2019, pp. 206–211.

[22] S. Roy, B. Stiene, S. K. Millican, and V. D. Agrawal, "Improved Pseudo-Random Fault Coverage Through Inversions: a Study on Test Point Architectures," *J. Electron. Test*, vol. 36, no. 1, p. 123–133, Feb. 2020.

[23] Y. Lecun, Y. Bengio, and G. Hinton, "Deep Learning," *Nature Cell Biology*, vol. 521, pp. 436–444, 2015.

[24] C. Nwankpa, W. Ijomah, A. Gachagan, and S. Marshall, "Activation Functions: Comparison of trends in Practice and Research for Deep Learning," *ArXiv*, vol. abs/1811.03378, 2018.

[25] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[26] R. Kemker, M. McClure, A. Abitino, T. Hayes, and C. Kanan, "Measuring Catastrophic Forgetting in Neural Networks," *eprint cs.AI arXiv:1708.02072*, 2017.

[27] F. Brglez and H. Fujiwara, "A Neutral Netlist of 10 Combinational Benchmark Circuits and a Targeted Translator in FORTRAN," *Proceedings of the IEEE Int. Symp. on Circ. and Sys. (ISCAS)*, pp. 677–692, June 1985.

[28] F. Corno, M. S. Reorda, and G. Squillero, "RT-Level ITC'99 Benchmarks and First ATPG Results," *IEEE Design & Test of Comp.*, vol. 17, pp. 44–53, Jul. 2000.

[29] B. M. Wilamowski, "Neural Network Architectures and Learning Algorithms," *IEEE Industrial Electronics Magazine*, vol. 3, no. 4, pp. 56–63, Dec. 2009.