# Monitoring the Health of Emerging Neural Network Accelerators with Cost-effective Concurrent Test

Qi Liu*, Tao Liu†, Zihao Liu†,Wujie Wen* and Chengmo Yang‡
*Lehigh University, †Florida International University, ‡University of Delaware
*{qil219, wuw219}@lehigh.edu, †{tliu023, zliu021}@fiu.edu, ‡chengmo@udel.edu

***Abstract*—ReRAM-based neural network accelerator is a promising solution to handle the memory- and computation-intensive deep learning workloads. However, it suffers from unique device errors. These errors can accumulate to massive levels during the run time and cause significant accuracy drop. It is crucial to obtain its fault status in real-time before any proper repair mechanism can be applied. However, calibrating such statistical information is non-trivial because of the need of a large number of test patterns, long test time, and high test coverage considering that complex errors may appear in million-to-billion weight parameters. In this paper, we leverage the concept of corner data that can significantly confuse the decision making of neural network model, as well as the training algorithm, to generate only a small set of test patterns that is tuned to be sensitive to different levels of error accumulation and accuracy loss. Experimental results show that our method can quickly and correctly report the fault status of a running accelerator, outperforming existing solutions in both detection efficiency and cost.**

## I. Introduction

Deep neural network (DNN) has nowadays become the state-of-the-art technique for many real-world applications, such as computer vision, speech recognition, robotics, gaming, healthcare and self-driving vehicles [1], [2]. To well leverage its superb cognitive capability, it is crucial to develop energy-efficient and high-performance accelerators to handle its non-traditional workloads that are both computationally intensive and memory intensive. Recently processing-in-memory (PIM) accelerators built upon emerging resistive random access memory (ReRAM or memristor) have become one of the most promising solutions, by integrating the memory and logic into one module to address the long-lasting "memory wall" problem in existing FPGA and ASIC accelerators. These memristor devices can be used both as weight memories and logic units to naturally perform the matrix-vector multiplication in parallel with zero cost in data movement, improving the computing efficiency by orders of magnitude over CMOS-based counterparts [3], [4].

Unfortunately, the well-trained DNN weight parameter, which is stored by the memristance or resistance value of each device, suffers from large uncertainty at both initial model deployment stage and accelerator in-use stage because of its unique physical limitations, e.g. cell-to-cell process variations, stochastic programming, random electrical or thermal noise, cycle-to-cycle resistance drifting and endurance issues during the run time [5], [6]. These complex errors, if left unchecked and accumulated in weights, can eventually lead to significant model accuracy loss when performing an on-device inference task after a certain time period. To rescue the erroneous accelerators, various repair mechanisms, including hardware redundancy, error correction, fault-aware remapping and cloud-edge collaborative model retraining, are proposed [7], [8]. These methods usually incur different overhead and are tailored for different stages based on the severity of the fault model, e.g. fault model that cannot be fixed by model remapping may need more expensive retraining at cloud. Therefore, it is essential to precisely and timely monitor the fault status of a running accelerators, so as to efficiently apply an appropriate repair solution.

A common approach is to design a few test inputs that are sensitive enough to calibrate the fault status (or fault detection) of such emerging DNN accelerators during the run timing, akin to test vectors used in logic and memory test. However, it suffers from the following challenges: **First**, a trustworthy fault status needs to be identified by a statistical manner which would involve a large number of test data, e.g. observing the inference accuracy drop of a well-trained model after feeding 10K$\sim$50K testing images. As a result, the process is both time-consuming and expensive. More crucially, such an excessive number of extra inference for test purpose only could incur further accuracy degradation given that device errors, such as endurance, can be also generated or further deteriorated during the run-time. The longer the test sequence is, the higher the probability that errors may accumulate during testing. **Second**, it is difficult to guarantee the sensitivity of test patterns, e.g. responding to any weight distortions that might lead to noticeable accuracy decay, because errors can be distributed among a large number of weights, while the number of test patterns is quite limited. For example, a recent study shows that testing an already fault accelerator with some normal testing data can also produce confidence scores very close to that of non-fault versions [9]. Such a minor difference, however, is unable to reflect its fault status because of the insufficient sensitivity of the selected test data, resulting in false negatives.

In response to these challenges, this paper makes a solid step towards developing high quality test patterns from an algorithm perspective, to accurately report the health status of emerging accelerators during the run time. The generated patterns have a very limited number but high sensitivity (coverage) to any tiny status change, thereby truly enabling low cost and fast concurrent test for these emerging DNN accelerators whose device technology is far less mature than CMOS and need post-fabrication self-testing and self-healing. Our major contributions are:

- We investigate the testing data to identify a few "corner data" which can easily confuse DNN's decision for testing purpose (namely "C-TP"). Compared with the state-of-the-art adversarial example-based test pattern ("AET"), we found that the obtained "C-TP" can produce much larger confidence score changes when encountering the same weight disturbance, indicating significantly enhanced fault status detection capability.
- To address the biased decision making of "C-TP" (initial weights are still biased to a certain class), thus unbalanced sensitivity to different weight errors, we propose to generate optimized test patterns ("O-TP") from scratch by leveraging DNN's back-propagation training algorithm. A small set of "white noise" patterns, e.g. only 10 for MNIST, are tuned to be sensitive to different levels of error accumulation and accuracy loss.
- Extensive experimental results show that both "O-TP" and "C-TP" can significantly outperform the latest "AET" method in fault detection by using confidence score changes. Moreover, "O-TP" needs the least number of patterns to achieve the best performance in monitoring the tiny accuracy status change.

## II. Background

### A. ReRAM Based Neural Network Accelerators

**Deep Neural Network (DNN)** relies on the complex multiple layer structure and learning algorithms to abstract the data at a high level [10]. A typical DNN topology usually consists of multiple convolutional (CONV), pooling and fully-connected (FC) layers. Among them, CONV layers abstract features from the inputs through the kernel-based convolutions, and FC layers further rank the confidence of each class based on weighted features and non-linear activation. The dot-production is the key operation in DNN computation.

**ReRAM based DNN Accelerator.** The value of DNN weights can be encoded as different conductance levels $g_{ij}$ on memristive cells. The inputs are converted, via a set of digital-to-analog (DAC) converters, to different voltage levels $v_i$ and are applied to the word-lines (WL). Through the crossbar structure, current $I_{ij}$ passing through the bit-line (BL) represents the dot-production of the input vector and the weights. The outputs are latched using Sample and Hold (S&H) circuits and then obtained via a set of analog-to-digital (ADC) converters. The parallel architecture and in-situ computing make the ReRAM crossbar a promising solution to accelerate DNN computation. For example, there emerges a variety of ReRAM based DNN accelerator designs, such as "Dot-product Engine" [11], "Prime" [12], "ISAAC" [4], "Pipelayer" [13], and "Time" [14].

### B. ReRAM Defects and Existing Solutions

ReRAM defects can distort the mapped weights (or conductance levels), thus compromising the system reliability and performance on ReRAM based DNN accelerators. Prior work [15], [16], [17] report that hard fault, induced by fabrication and endurance limitations, occurs when a ReRAM device freezes itself in a low resistance state (LRS) or high resistance state (HRS), resulting in the stuck-at-one (SA1) fault or stuck-at-zero (SA0) fault that can significantly degrade the accuracy. The resistance drift [18], induced by the change of resistance level over time, can lead to soft errors when accelerator is in use. Stochastic variations, such as random telegraph noise [19], and programming uncertainty [7], cause deviations in DNN weights from their nominal values. For example, programming uncertainty can be formulated as $w' \leftarrow w \cdot e^{\theta}$ *s.t.* $\theta \sim N(0, \sigma^2)$ where $w'$ is the neural network parameter with programming errors due to the memristor resistance variation $\theta$, which follows a log-normal distribution. To address these defects in DNN accelerators, existing solutions include error-correcting circuit [20], fault-tolerant training [8] and redundancy based re-mapping [7]. Besides, algorithm-level solutions [17] can be also used to better enhance the error resilience on ReRAM accelerators. Asides from these repair schemes, fault detection methods, such as recent adversarial example testing [9], are required in prior to analyze the existence of ReRAM defects and error patterns. Our work belongs to this category and we select programming variation and random soft error as two examples to demonstrate the effectiveness of our proposed testing pattern generating method.

## III. Cost-effective Concurrent Test

In this section, we propose two test pattern generation approaches to test the fault status of the running PIM accelerator. Our goal is to develop a small set of test patterns, i.e. special images in regular training or inference datasets or images not belonging to them at all, such that observing obvious inference result change (e.g. via output confidence score [9]) of these patterns will be sufficient for determining the fault status of a running accelerator. Overall, a high-quality set of test patterns should fulfill two requirements: **1) the fault coverage should be sufficient, meaning that these patterns need to be sensitive enough to respond to any weight distortions that might lead to prominent accuracy drop**, and **2) The number of the test pattern is much less than that of original test images for a fast and low-cost testing.** We aim to explore how to generate the test pattern that satisfies those two requirements simultaneously.

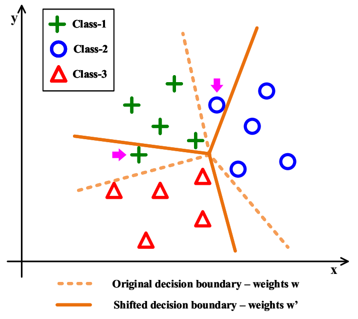
Class-1
Class-2
Class-3
y
x
Original decision boundary – weights w
Shifted decision boundary – weights w'

Fig. 1: Selecting "corner data" sensitive to the decision boundary change

### A. Corner Data as Test Pattern ("C-TP")

Our first approach is to directly use some existing "corner data" inputs (namely "C-TP", abstract from corner data-based test pattern), whose inference results are more vulnerable to the uncertainty of neural network models, e.g. weight distortions. Assume $f_w(\cdot) : X \rightarrow Y$ represents a well-trained model with input $X \in \mathbb{R}^d$, output $Y \in \mathbb{R}^n$ and weight $w$ store in ReRAM cells, and $f_{w'}(\cdot)$ is the shifted model with weight errors ($w \rightarrow w'$). Fig. 1 illustrates an example of the decision boundary change from $f_w(\cdot)$ to $f_{w'}(\cdot)$ due to the weight errors. The inference results of the two "corner data" as a result of weight errors, are changed from class-1 to class-3 and class-2 to class-1, respectively. This approach requires quantitatively measuring the distances between inputs to the original decision boundaries, which is difficult for a high-dimensional classification problem. To avoid modeling the decision boundary, we consider leveraging the *confidence difference* among different classes when testing an image, to select the "C-TP". For example, to create a decision change from class-$i$ to class-$j$ for the original trained model, we select the image $X$ with the least confidence difference, i.e. $\min\left(Z(X)_i - Z(X)_j,\ j \neq i\right)$, here $Z(X)_i$ is the logits value representing the output confidence of class $i$ for input $X$. However, in order to satisfy high coverage, the number of this kind of "corner data" with two class change is a lot. For an $n$-class classification problem, there will be at least $\frac{n(n-1)}{2}$. Apparently, this solution can not satisfy the second requirement: the number of patterns should be small.

To address this problem, we assume any ideal "C-TP" available in a multidimensional space should have the following nature: for test data $X$, the distance from $X$ to all decision surfaces should be the same. It means $X$ has an equal probability of crossing any decision surface so that class is changed without any bias whenever any weight error occurs. However, it is very difficult to identify such data in input space $X$ explicitly considering the difficulty of modeling the high dimensional decision boundary. Instead, we use the following rule to select "C-TP": $\min\left(\sqrt{\frac{1}{n}\sum_i^n (Z(X)_i - \overline{Z(X)})^2}\right)$, a.k.a minimize the standard deviation of output logits. We directly search "C-TP" in the inference dataset by sorting the standard deviation of the output logits (from small to large) and then select the top-$m$ "corner data" as "C-TP". Note in an ideal case, $m$ can be as small as the number of classes $n$. However, since there may not exist ideal "corner-data" in the inference dataset that has the same distance to all decision surfaces, in other words, such selected "C-TP" is still biased during decision making, which may not have the same sensitivity to different

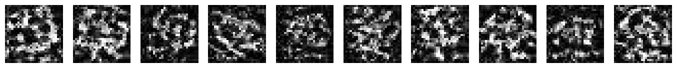
Fig. 2: 10 "O-TP" (white noise style) test patterns generated from LeNet-5 and MNIST.

weight errors. To get better fault coverage, we choose $m \geq n$, e.g. $O(n)$ instead of the original $O(n^2)$ to reduce the number of needed "C-TP". We will analyze it in our experiments in section IV.

### *B. Generate Test Pattern through Optimization Algorithm ("O-TP")*

As discussed in Section III-A, "C-TP" may exhibit very different sensitivity levels as weight errors change due to the biased decision making of such data in the clean model. To further improve fault coverage and lower the number of needed patterns, we propose to generate the test patterns from scratch, inspired by the training process of a neural network model with an optimization algorithm (namely "O-TP", abstract from optimization-based test pattern). Specifically, the training process starts with randomly initialized weight parameters $w$ and iteratively updates them by minimizing a given loss function $L$ until reaching the convergence. Since the input and weights are interchangeable in terms of inference results, in our case, we attempt to create a set of test patterns from a group of randomly initialized input $X^{TP}$, and then iteratively update $X^{TP}$ towards two goals: *(1) $f_w$ (the originally trained model) is extremely confused about $X^{TP}$, i.e., random noise with almost equal probability for each class. This ensures that $X^{TP}$ does not have any bias to certain weights in $f_w$ and can freely respond to any weight errors that may happen. (2) $f_{w'}$ (the model with its weights suffering errors) is very confident (e.g. $\sim$ 100%) about the prediction of $X^{TP}$ for a certain class $i$. The biased decision indicates that the accumulated errors in certain weights start to impact the inference accuracy of the accelerators.* To satisfy both targets, we propose to minimize the following cross-entropy loss function:

$$\arg\min_{X^{TP}} -\left(\alpha \cdot \sum_{i=1}^{n} l_i log(f_w(X^{TP})) + (1-\alpha) \cdot \sum_{i=1}^{n} l_i^{'} log(f_{w'}(X^{TP}))\right) \quad (1)$$

where $l_i$ ($0 \leq i \leq n$) are the soft labels with equal confidence over all classes, and $l'_i$ ($0 \leq i \leq n$) are the hard labels that are sensitive to a specific class. Here $\alpha \in (0,1)$ is a coefficient, which indicates the relative importance of first and second terms. The first and second terms represent the constraints for (1) and (2), respectively. This minimization problem can be solved with algorithms such as stochastic gradient descent. The detail of the algorithm is shown in Algorithm 1). In line 16, we use standard deviation of output in original model $f_w(\cdot)$ ($std$ is a standard deviation function defined in line 10) and $L_1$ distance between the outputs of error model $f_{w'}(\cdot)$ and target $T$ as the constraints of the two terms, respectively. Here, $\epsilon_1 \in (0,1)$ and $\epsilon_2 \in (0,1)$ are constraint coefficients. To find a high-quality test pattern, $\epsilon_1$ and $\epsilon_2$ should be very small, e.g. $1e-3$. To increase test coverage, we explore the number of patterns needed for each class. Thus, for a $n$-class classification problem, the number of "O-TP" should be $n$ in the ideal case. We expect that "O-TP" requires a fewer number than "C-TP" for the same high coverage. To be conservative, we still consider that on average $k$ patterns are needed for each class. The number of "O-TP" should be $k \cdot n$. In our experiments (section IV), we find that the needed "O-TP" in general will be equal to the number of class (i.e. $k = 1$). Fig.2 visualizes 10 "O-TP" for the 10-class MNIST dataset (handwritten digits). We can observe that such generated test patterns are completely different from the input images used in training and testing.

**Algorithm 1: Optimized Test Pattern Generating**

```
   Data:
   $f_w(.) \in \mathbb{R}^n$;  // the DNN model
1  $f_{w'}(.) \in \mathbb{R}^n$;  // the DNN model with weight error
2  $X$;  // the input image with random noise, i.e. $X \in \mathcal{N}(0, 0.1)$
3  $min_X$ , $max_X$   // the boundary of the input image
4  $T$;  // target vector for $f_{w'}(.)$
5  $\Gamma = \{l_1, l_2, \dots, l_n\}$;  // the soft label set with equal confidence, i.e.
     $\{\frac{1}{n}, \frac{1}{n}, \dots, \frac{1}{n}\}$
6  $\Gamma' = \{l'_1, l'_2, \dots, l'_n\}$;  // the hard label set over specific class i.e.
     $\{1, 0, \dots, 0\}$
7  M, lr;  // the maximal iterations and learning rate
8  $\epsilon_1, \epsilon_2$;  // the constraint coefficients for two terms respectively
   Result:
   $X^{TP}$;  // the test pattern
9  $X^{TP} \leftarrow X$ , $m \leftarrow 0$
10 $mean(X) \leftarrow \frac{1}{n}\sum_i^n f_i(X)$ ,
     $std(X) \leftarrow \sqrt{\frac{1}{n}\sum_j^n (f_j(X) - mean(X))^2}$
11 $loss(X) \leftarrow -\left(\sum_i^n l_i log(f_w(X)) + \sum_i^n l'_i log(f_{w'}(X))\right)$
12 while $m < M$ do
13     $g = \nabla_{X^{TP}} loss(X^{TP})$
14     $X^{TP} = X^{TP} - lr * g$
15     clip($X^{TP}$, $min_X$, $max_X$)
16     if $std(X^{TP}) < \epsilon_1$ and $\|f_{w'}(X^{TP}) - T\|^1 < \epsilon_2$ then
17         break;
18     m+=1
19 return $X^{TP}$
```

## IV. Evaluation

### *A. Experimental Setup*

**Benchmark.** We evaluate our methods on both MNIST and CIFAR10 datasets. MNIST is gray-scale handwritten digits, consisting of 60K $28 \times 28$ training images and 10K testing images. CIFAR10 is a $32 \times 32$ color images dataset with 10 classes, including 50K training images and 10K testing images. "LeNet-5" model [21] is adopted to train and test the MNIST dataset, and a customized 7-layer neural network "ConvNet-7", which consists of 4 convolutional layers and 3 fully connected layers, is used for CIFAR10 dataset. The accuracy of the well-trained LeNet-5 (MNIST) and ConvNet-7 (CIFAR10) are 99.04% and 81.62%, respectively.

**Error Model.** Two error models are included in our evaluation. The weight variation induced by programming error, follows a log-normal distribution $w' \leftarrow w \cdot e^{\theta}$ *s.t.* $\theta \sim N(0, \sigma^2)$, where $\sigma$ is the noise intensity. The distorted model is presented as $f_{w'(\sigma)}$. We also inject the random soft errors into models during the run time, with $f_{w'(p)}$ representing the fault model with such random errors. Here $p$ is the error probability.

For programming error, we create the LeNet-5 (ConvNet-7) fault models with different levels of variance $\sigma = \{0.05 \rightarrow 0.5\}$ ($\sigma = \{0.05 \rightarrow 0.3\}$). We construct 100 fault models for each sampled $\sigma$, i.e., 1k (600) fault models for LeNet-5 (ConvNet-7). The errors are randomly injected into DNN weights across all layers. TABLE I and TABLE II report the average accuracy degradation on fault models with different $\sigma$ for LeNet-5 and ConvNet-7, respectively. For random soft errors, we select p=0.5% and p=1% on LeNet-5, which degrades the original accuracy from 99.04% to 95.36% (p=0.5%) and 87.47% (p=1%), respectively. On ConvNet-7, we select p=0.1% and p=0.3%, which reduces the original accuracy from 81.62% to 77.17% (p=0.1%) and 67.56% (p=0.3%), respectively.

**Test Pattern.** For our *C-TP (corner data based test pattern)* method, we selected 50 corner data as test patterns on each dataset. For our *O-TP (optimized test pattern)* method, we apply $\alpha = 0.5$ in Eq. 1 to indicate the equal importance of each term in the

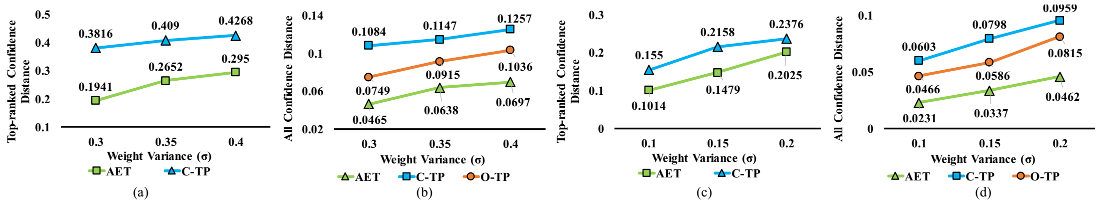

Fig. 3: Top-ranked confidence distance and all confidence distance of test data under AET, C-TP, O-TP on LeNet-5 (MNIST) and ConvNet-7 (CIFAR10) with programming error. (a)(b) LeNet-5. (c)(d) ConvNet-7.

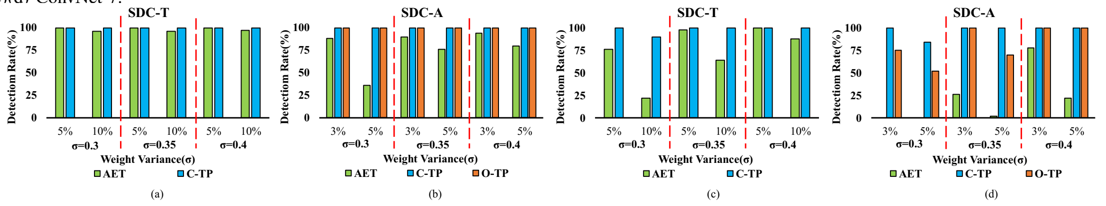

Fig. 4: The detection rate of test data under AET, C-TP, O-TP on LeNet-5 (MNIST) and ConvNet-7 (CIFAR10) with programming error on SDC-T (SDC-T5% and SDC-T10%) and SDC-A (SDC-A3% and SDC-A5%) measurements. (a)(b) LeNet-5. (c)(d) ConvNet-7.

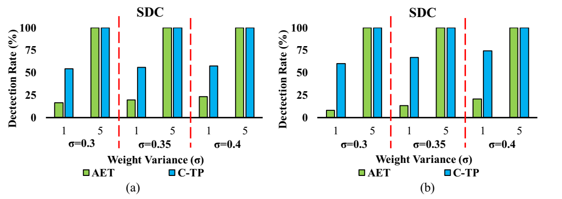

Fig. 5: The detection rate of test data under AET, C-TP, O-TP on LeNet-5 (MNIST) and ConvNet-7 (CIFAR10) with programming error on SDC-1 and SDC-5 measurements. (a) LeNet-5. (b) ConvNet-7.

optimization process. $\epsilon_1 = 1e-3$ and $\epsilon_2 = 1e-3$ are selected for each dataset. We compare our C-TP and O-TP with the state-of-the-art *AET (adversarial example based testing)* method [9], which uses adversarial examples (i.e. FGSM) as testing images based on the fact that they are more sensitive to weight variance than normal inputs. For a fair comparison, we use the same number (i.e., 50) of testing images to detect the fault model for each evaluated method.

**Metrics.** We adopt the similar metrics from [22] in our evaluation: *SDC-1* and *SDC-5*: compare the top-1/top-5 class between the target model and the ideal model. A difference indicates target model is faulty. *SDC-T5%* and *SDC-T10%*: compare top-ranked confidence score between the target model and the ideal model. A difference more than ±5% or ±10% indicates the target model is faulty. In addition, we create two new detection criteria since our O-TP method does not assess the top-ranked class, i.e., *SDC-A3%* and *SDC-A5%*–compare the average confidence distance between target model and ideal model. A difference of more than ±3% or ±5% indicates the target model is faulty. Such class change and confidence score distance can be used to measure the performance of proposed methods and existing testing patterns. Note that the objective of our proposed methods (as well as the state-of-the-art AET) is to create effective test patterns to enlarge the confidence distance between the fault model and the ideal model. Also, we follow [9] to use the *detection rate*, i.e., $^{\text{the number of detected fault models}}/_{\text{total number of fault models}}$, to show the fault detection performance of evaluated methods.

## B. Results and Analysis

**Detection Effectiveness.** Fig. 3 shows our observation on the confidence distance, including top-ranked confidence distance (SDC-T) and averaged all confidence distance (SDC-A) on LeNet-5 and ConvNet-7. As shown in Fig.3, our proposed C-TP method (O-TP) achieves a larger top-ranked (all) confidence distance than that of AET. Further, we evaluate the detection rate to verify their performance in detecting the fault model. As shown in Fig. 4(a) and (b), our proposed C-TP and O-TP methods can both achieve the 100% detection rate on SDC-T5%, SDC-T10%, SDC-A3%, SDC-A5% on the LeNet-5 fault models. In contrast, the AET method is less effective than our methods, especially for the smaller error $\sigma$ = 0.3 (i.e., 88% for SDC-A3% and 36% for SDC-A5%). As shown in Fig.4(c) and (d), more significant performance gap between our proposed methods and the AET on ConvNet-7 fault models for more complex CIFAR10 dataset. In particular, as shown in Fig. 4 (d), the detection rate of AET is close to 0% on SDC-A3% and SDC-A5% for small errors $\sigma = 0.3$. Note that it is more difficult to detect such smaller errors. The AET method is invalid to detect the fault models with such tiny errors ($\sigma < 0.3$) with SDC-A criteria. Fig. 5 further shows the detection rates measured by top-1/top-5 class. We can observe that the proposed C-TP method still performs better than AET on different $\sigma$ on SDC-1 on MNIST and CIFAR10. TABLE III further lists the average detection rate based on different measurements for all different $\sigma$. For all detection criteria, our C-TP and O-TP achieve a better detection rate than AET. In particular, the C-TP method achieves a higher detection rate than the optimized O-TP, due to its higher confidence score distances than O-TP. This is because C-TP is more biased than "white noise" like O-TP for decision making in initial clean models. However, this does not mean C-TP is better than O-TP to monitor accuracy drop, as we shall show in Fig 8. The proposed C-TP and AET both can achieve 100% detection rate with SDC-5 measurement. This is because in the 10-class MNIST and CIFAR10 datasets, top-5 is easily changed when weight variance occurs. Fig. 6 show our results with random errors.

TABLE I: The accuracy of original LeNet-5 model and fault LeNet-5 models $f_{w'(\sigma)}$ with different $\sigma$ on MNIST

| weight error ($\sigma$) | 0 (original) | 0.05 | 0.1 | 0.15 | 0.2 | 0.25 | 0.3 | 0.35 | 0.4 | 0.45 | 0.5 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| LeNet-5 accuracy(%) | 99.04% | 99.01% | 98.87% | 98.64% | 98.26% | 97.48% | 96.02% | 94.7% | 91.73% | 89.32% | 86.61% |

TABLE II: The accuracy of original ConvNet-7 model and fault ConvNet-7 models $f_{w'(\sigma)}$ with different $\sigma$ on CIFAR10

| weight error ($\sigma$) | 0 (original) | 0.05 | 0.1 | 0.15 | 0.2 | 0.25 | 0.3 |
|---|---|---|---|---|---|---|---|
| ConvNet-7 accuracy(%) | 81.62% | 81.33% | 80.51% | 79% | 75.75% | 72.37% | 66.59% |

TABLE III: The average detection rate of test data under AET, C-TP, O-TP on LeNet-5 and ConvNet-7 with programming error (all $\sigma$)

| | LeNet-5 (MNIST) | | | | | |
|---|---|---|---|---|---|---|
| | SDC-1 | SDC-5 | SDC-T5% | SDC-T10% | SDC-A3% | SDC-A5% |
| AET | 19.2% | 100% | 94.5% | 77.1% | 71.6 % | 49.7% |
| C-TP | 56.8% | 100% | 100% | 100% | 100% | 94.8% |
| O-TP | - | - | - | - | 84.4% | 71.9% |
| | ConvNet-7 (CIFAR10) | | | | | |
| AET | 13.7% | 100% | 91.3% | 58.4% | 34.7% | 8.1% |
| C-TP | 67.2% | 100% | 100% | 96.7% | 100% | 94.7% |
| O-TP | - | - | - | - | 91.7% | 74% |

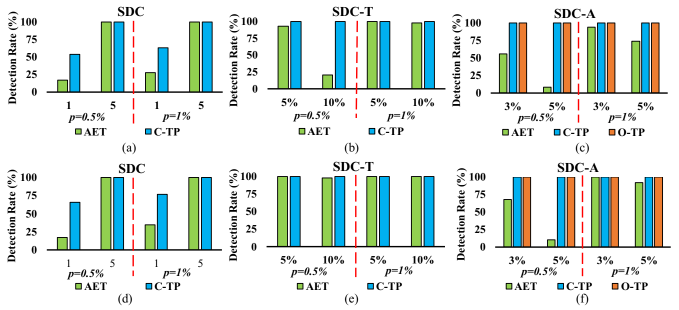

Fig. 6: The detection rate of test data under AET, C-TP, O-TP on LeNet-5 (MNIST) and ConvNet-7 (CIFAR10) with random errors on 6 different detection measurements. (a)(b)(c) LeNet-5 with MNIST. (d)(e)(f) ConvNet-7 with CIFAR10.

Our two methods can also achieve a higher detection rate than AET across all selected detection criteria on MNIST and CIFAR10.

**Sensitivity Analysis.** In our sensitivity analysis, we try to answer the question: *Why our methods are more sensitive to the weight variance than AET?* First, in the AET method, the FGSM-based adversarial examples are used to mislead the decision by adding perturbations into the input. Therefore the confidence vector of original model $f(x,w)$ will be significantly changed to $f(x',w)$, thus a large confidence distance between $f(x,w)$ and $f(x',w)$ can be expected for all/top confidence distance. However, it can not guarantee that $f(x',w)$ and $f(x',w')$ have a detectable confidence distance when weight variance occurs. For the detected cases with AET, since DNN model is already fooled by adversarial examples, **it is highly possible that DNN maintains low confidence for each class** (e.g., for a 3-class classification, confidence changes from (0.8,0.1,0.1) to (0.3,0.4,0.3)). **Therefore, a tiny weight variance could accidentally change the confidence score, which can be detected later**. In contrast, our proposed methods are quite different. We intentionally develop these test patterns. For C-TP, we select those "corner data" with the smaller standard deviation of "soft confidence score". They can easily cross the decision surface if any small weight error occurs, thus enlarging the confidence distance between the original model and the fault model. O-TP uses a specific soft label (i.e., target confidence 0.1 for each class in a 10-class classification) as a target label for original model and uses a hard label for the fault model to generate the test pattern, by using gradient descent optimization. When weight error occurs, the "soft confidence" can approach the "hard confidence", therefore creating a larger confidence distance. This indicates that our methods are more sensitive to weight variance than AET.

**Stability Analysis.** We now analyze the stability of the proposed methods. A good testing method is expected to maintain reliable detection performance on any fault model. The stability of the testing method can be measured by the coefficient of variation (CV), which is represented as $\frac{\sigma}{\mu}$, where $\sigma$ and $\mu$ are the standard deviation and mean of the confidence distance between clean and fault models. A smaller value indicates better stability (non-variability). As TABLE IV shows, our C-TP and O-TP are more stable than AET. This is because our two methods select and generate the test data with a similar characteristic (i.e., the "corner data"), while AET randomly selects test patterns from the testing dataset and adds FGSM-based adversarial perturbations in a coarse-grained manner.

**Efficiency.** We now evaluate the efficiency of the proposed methods. On ReRAM-based DNN accelerator, we aim to detect the fault model by using as few as test patterns as possible. Note that for a fair comparison, we use 50 test images for each method. However, our proposed O-TP does need so many test images in fact. We use standard deviation to better measure how many test images are needed for the three methods. As shown in Fig.7, to achieve the same level of standard deviation, our methods consume a less number of test data than AET on both MNIST and CIFAR10. In particular, Fig. 7 (a) shows, for a reliable detection (i.e., standard deviation has converged to a certain level), AET requires at least 150 test images, while our proposed C-TP method needs only 50 test images. As shown in Fig.7 (b), our O-TP can always maintain the stable detection even with very few (i.e. 10) test images, as such patterns are well optimized to enlarge the confidence distance. We can observe the similar result on CIFAR10 (see Fig. 7 (c) and (d)). For the time cost of pattern generation, since our O-TP and C-TP, which is similar to AET, only requires to generate once at the cloud, so it is very low-cost.

**Accuracy Status Testing and Analysis.** Fig. 8 shows the relationship between confidence distance and accuracy of the fault model, which is tested with original testing images, AET, C-TP and O-TP patterns. The blue bars indicate model accuracy and the lines represent the confidence distances under different $\sigma$. First, a range of confidence distance can be observed for each type of testing pattern. For the original testing image (randomly selected from the test dataset), the range of confidence distance is merely from 0 to $\sim$ 0.01. This means that using original images to distinguish the different accuracy degradation is very difficult due to the small range of confidence distance change. AET is better than original images, whose range of confidence distance is from 0 to $\sim$ 0.04. An ideal testing method is expected to maintain a large range of confidence distance but a higher variance (or slope). We use 0.01 as the confidence variance unit to evaluate the testing methods. For our C-TP and O-TP, they can increase the range of confidence variance from 4 levels $\{0.01 \rightarrow 0.04\}$ in AET, to 11 levels ($\{0.01 \rightarrow 0.11\}$) in C-TP and 10 levels in O-TP. It is expected that the method which maintains a linear trend on its confidence variance change with an enlarged range, can better correlate with accuracy change. Therefore, our O-TP performs better than C-TP. We can observe that while C-

TABLE IV: The coefficient of variation (CV) of confidence distance for AET, C-TP, O-TP on LeNet-5 with various weight errors ($\sigma$)

| weight variance ($\sigma$) | 0.05 | 0.1 | 0.15 | 0.2 | 0.25 | 0.3 | 0.35 | 0.4 | 0.45 | 0.5 |
|---|---|---|---|---|---|---|---|---|---|---|
| AET | 0.38 | 0.39 | 0.37 | 0.33 | 0.35 | 0.34 | 0.35 | 0.31 | 0.30 | 0.30 |
| C-TP | 0.17 | 0.16 | 0.15 | 0.16 | 0.16 | 0.15 | 0.16 | 0.13 | 0.12 | 0.13 |
| O-TP | 0.16 | 0.15 | 0.15 | 0.17 | 0.15 | 0.16 | 0.14 | 0.15 | 0.13 | 0.12 |

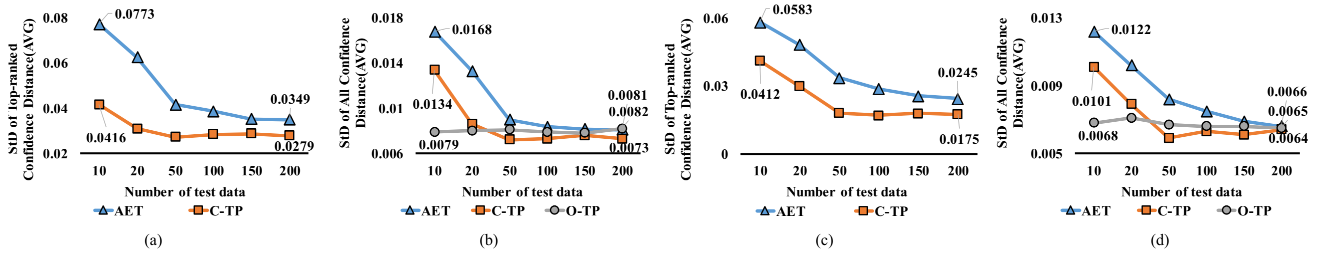

Fig. 7: The standard deviation of all/top-ranked confidence distance w.r.t. different number of test data under AET, C-TP, O-TP on MNIST/CIFAR10. (a)(b) MNIST. (c)(d) CIFAR10.

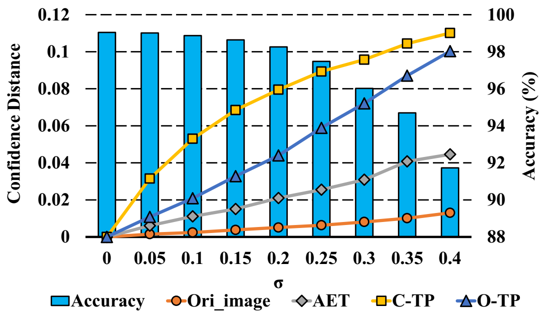

Fig. 8: The relationship between confidence distance and model accuracy with weight error $\sigma$

TP has a larger slope with $\sigma = 0.05, 0.1$, the accuracy change can be neglected (from 99.04% to 99.01% and 98.87% respectively). In contrast, for a prominent accuracy degradation ($\sigma = 0.3 - 0.4$), C-TP has only two levels of confidence distance to capture the sharp accuracy change (accuracy drop from 96.02% to 91.73%). On the other hand, for the same accuracy degradation in $\sigma = 0.3 - 0.4$, O-TP offers four levels of confidence distance to detect such a change more accurately and consistently due to a higher slope than C-TP. This means that O-TP offers better testing effectiveness than C-TP, while C-TP can be better than AET and testing with original images.

## V. Conclusion

In this work, we propose two types of test patterns to monitor the health status of emerging DNN accelerators. We found that high-quality test pattern needs to satisfy two requirements: high coverage and low-cost. We elaborately select "corner data" that can easily cross the decision boundary as the first method to create test pattern ("C-TP"). To further improve fault detection performance and coverage, we also design the second method ("O-TP")–a gradient descent based optimization algorithm to generate test patterns from scratch. In our experiment, we evaluate both methods through a variety of weight error settings, to characterize their performance on detecting fault models and their accuracy status. Overall, both methods outperform the latest adversarial example-based test pattern ("AET"). For fault detection, our "C-TP" achieves a higher detection rate than "O-TP" because of a higher sensitivity to confidence score change incurred by weight errors, while "O-TP" requires fewer number of patterns. For the accuracy status change detection, our "O-TP" delivers a better accurate match between confidence distance and accuracy degradation than "C-TP".

## Acknowledgments

This work was supported in part by NSF Grants SHF-2011236, SPX-2006748 and SHF-1909854.

## References

[1] C. Szegedy, "An overview of deep learning," *AITP 2016*, 2016.

[2] D. Silver *et al.*, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[3] Y. Chen *et al.*, "Diannao family: energy-efficient hardware accelerators for machine learning," *Communications of the ACM*, vol. 59, no. 11, pp. 105–112, 2016.

[4] A. Shafiee *et al.*, "Isaac: A convolutional neural network accelerator with in-situ analog arithmetic in crossbars," *ACM SIGARCH Computer Architecture News*, vol. 44, no. 3, pp. 14–26, 2016.

[5] G. Medeiros-Ribeiro *et al.*, "Lognormal switching times for titanium dioxide bipolar memristors: origin and resolution," *Nanotechnology*, vol. 22, no. 9, p. 095702, 2011.

[6] T. Chang *et al.*, "Short-term memory to long-term memory transition in a nanoscale memristor," *ACS nano*, vol. 5, no. 9, pp. 7669–7676, 2011.

[7] L. Chen *et al.*, "Accelerator-friendly neural-network training: Learning variations and defects in rram crossbar," in *Proceedings of the Conference on Design, Automation & Test in Europe*. European Design and Automation Association, 2017, pp. 19–24.

[8] C. Liu *et al.*, "Rescuing memristor-based neuromorphic design with high defects," in *2017 54th ACM/EDAC/IEEE Design Automation Conference (DAC)*. IEEE, 2017, pp. 1–6.

[9] H.L.X.L. Wen Li, Ying Wang, "Rramedy: Protecting reram-based neural network from permanent and soft faults during its lifetime," in *IEEE International Conference on Computer Design (ICCD)*. IEEE, 2019.

[10] G.E. Hinton *et al.*, "Reducing the dimensionality of data with neural networks," *science*, vol. 313, no. 5786, pp. 504–507, 2006.

[11] M. Hu *et al.*, "Dot-product engine for neuromorphic computing: Programming 1t1m crossbar to accelerate matrix-vector multiplication," in *Proceedings of the 53rd annual design automation conference*. ACM, 2016, p. 19.

[12] P. Chi *et al.*, "Prime: A novel processing-in-memory architecture for neural network computation in reram-based main memory," in *ACM SIGARCH Computer Architecture News*, vol. 44, no. 3. IEEE Press, 2016, pp. 27–39.

[13] L. Song *et al.*, "Pipelayer: A pipelined reram-based accelerator for deep learning," in *2017 IEEE International Symposium on High Performance Computer Architecture (HPCA)*. IEEE, 2017, pp. 541–552.

[14] M. Cheng *et al.*, "Time: A training-in-memory architecture for memristor-based deep neural networks," in *Proceedings of the 54th Annual Design Automation Conference 2017*. ACM, 2017, p. 26.

[15] C.Y. Chen *et al.*, "Rram defect modeling and failure analysis based on march test and a novel squeeze-search scheme," *IEEE Transactions on Computers*, vol. 64, no. 1, pp. 180–190, 2014.

[16] C. Xu *et al.*, "Overcoming the challenges of crossbar resistive memory architectures," in *2015 IEEE 21st International Symposium on High Performance Computer Architecture (HPCA)*. IEEE, 2015, pp. 476–488.

[17] T. Liu *et al.*, "A fault-tolerant neural network architecture," in *Proceedings of the 56th Annual Design Automation Conference 2019*. ACM, 2019, p. 55.

[18] B. Li *et al.*, "Ice: inline calibration for memristor crossbar-based computing engine," in *2014 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2014, pp. 1–4.

[19] F.M. Puglisi *et al.*, "A complete statistical investigation of rtn in hfo 2-based rram in high resistive state," *IEEE Transactions on Electron Devices*, vol. 62, no. 8, pp. 2606–2613, 2015.

[20] B. Feinberg *et al.*, "Making memristive neural network accelerators reliable," in *2018 IEEE International Symposium on High Performance Computer Architecture (HPCA)*. IEEE, 2018, pp. 52–65.

[21] Y. LeCun *et al.*, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[22] G. Li *et al.*, "Understanding error propagation in deep learning neural network (dnn) accelerators and applications," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. ACM, 2017, p. 8.