EGIS: Entropy Guided Image Synthesis for Dataset-Agnostic Testing of RRAM-Based DNNs

Anurup Saha, Chandramouli Amarnath, Kwondo Ma and Abhijit Chatterjee
School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA, USA
asaha74@gatech.edu, chandamarnath@gatech.edu, kma64@gatech.edu, abhijit.chatterjee@ece.gatech.edu

Abstract—**While resistive random access memory (RRAM) based deep neural networks (DNN) are important for low-power inference in IoT and edge applications, they are vulnerable to the effects of manufacturing process variations that degrade their performance (classification accuracy). However, to test the same post-manufacture, the (image) dataset used to train the associated machine learning applications may not be available to the RRAM crossbar manufacturer for privacy reasons. As such, the performance of DNNs needs to be assessed with carefully crafted dataset-agnostic synthetic test images that expose anomalies in the crossbar manufacturing process to the maximum extent possible. In this work, we propose a dataset-agnostic post-manufacture testing framework for RRAM-based DNNs using Entropy Guided Image Synthesis (EGIS). We first create a synthetic image dataset such that the DNN outputs corresponding to the synthetic images minimize an entropy-based loss metric. Next, a small subset (consisting of 10-20 images) of the synthetic image dataset, called the compact image dataset, is created to expedite testing. The response of the device under test (DUT) to the compact image dataset is passed to a machine learning based outlier detector for pass/fail labeling of the DUT. It is seen that the test accuracy using such synthetic test images is very close to that of contemporary test methods.**

I. Introduction

In recent years, resistive random access memory (RRAM) based *mixed-signal DNN accelerators* have emerged as a promising alternative to fully digital accelerators because of low power consumption, high density and compatibility with CMOS technology of RRAM crossbars [1], [2]. The RRAM based DNN chips store DNN weights using conductances of RRAM cells within crossbars. However, the conductances of RRAM devices are vulnerable to manufacturing process variations resulting in performance loss (as measured by classification accuracy). This necessitates careful performance screening of manufactured devices before field deployment.

Functional testing of DNN accelerators has been investigated in the past [3]–[5]. Further, testing of process variation induced performance (classification accuracy) degradation of RRAM based DNN accelerators has been addressed in [6], [7]. A major limitation of these works is that the test frameworks require access to the training/testing dataset of the DNN.

In this research, we ask a fundamental question: *Is it necessary to have access to the training or testing dataset of a DNN to develop functional tests for DNN hardware*? Such access becomes difficult in privacy-critical applications [8] such as in healthcare, robotics, and defense, where patient data in healthcare for example, cannot be divulged to third parties for privacy reasons. However, model information (such as DNN weights) can be made available to the hardware manufacturer. In such scenarios, dataset-agnostic test methodologies for DNNs are necessary for pass/fail manufacturing test of DNNs without access to the DNN training/testing dataset. We show in this research, that such dataset agnostic testing methods of quality comparable to existing test techniques are indeed possible and deployable using synthetic test datasets (images).

We first create a set of *purely synthetic images* using entropy guided image synthesis (EGIS). Next, we derive a compact image dataset from the generated synthetic images using image set compaction. During testing, a response signature is extracted from each DNN by applying the compact image dataset and the signature is passed to a trained outlier detector for pass/fail prediction. While prior work has used generative adversarial network (GAN) to augment test patterns [9] , training a GAN requires access to the DNN training dataset. To the best of our knowledge, this is the first work that develops test images *without access to the original DNN training or testing dataset*. In summary, we make the following contributions:
(1) We propose a novel dataset-agnostic image synthesis algorithm called entropy guided image synthesis (EGIS), suitable for privacy-critical applications, which generates synthetic images for RRAM-based DNN testing.
(2) An image set compaction algorithm is developed that allows DNN testing with 10-20 synthetic test images, significantly reducing test cost.
(3) Experiments show that the proposed dataset-agnostic testing framework achieves comparable test accuracy with respect to the state-of-the-art test methodologies that require access to the testing/training dataset of the DNN.

II. Overview

In this section, we explain the dataset privacy model assumed in this work and the overall testing approach.
Dataset privacy model: We assume that the end user and the DNN (RRAM crossbar) manufacturer are different entities. The end user has access to the training and testing datasets and is responsible for training the DNN. In our methodology, the end user or organization responsible for training the DNN needs to label D RRAM based manufactured DNNs as “pass” or “fail” and needs to provide the labels to the RRAM manufacturer. This could be from use of prior test algorithms [3]. The RRAM manufacturer independently designs a compact image dataset which is applied to a manufactured device under

978-3-9826741-0-0/DATE25/© 2025 EDAA

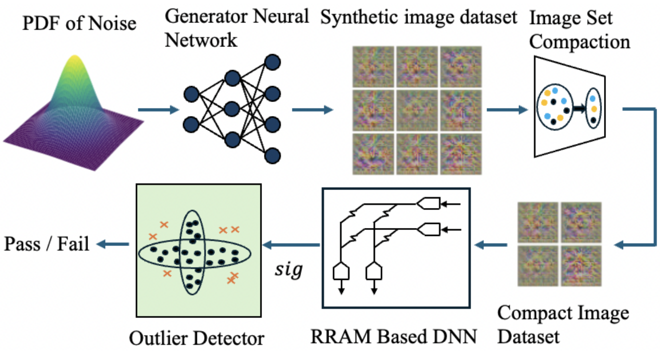

Fig. 1: Overall Framework

test (DUT) during post-manufacture test. The RRAM crossbar manufacturer also uses the labeled DUTs to design an outlier detector for pass/fail classification of a DUT. Since labeling of DUTs can be expensive, the proposed test framework is developed with only 1000 labeled DUTs, which is 3-4 orders fewer than the number of weights of the DNN (typically 1-100 million weights).

Overall testing approach: In the presence of manufacturing process variations, the accuracy of every manufactured DUT varies from its nominal accuracy. For a fixed acceptable accuracy threshold A_{cutoff} and a DUT with accuracy A, the test framework aims at predicting whether $A > A_{cutoff}$. The overall test framework, as shown in Fig. 1 involves three steps:

(1) A synthetic image dataset consisting of N images is generated using a generator neural network, which is trained using an entropy based loss functions to promote diversity within the synthetic image dataset.

(2) To enable post-manufacture testing using fewer images, we down-select K images from the N images above and create a compact image dataset ($K << N$). The image set compaction is achieved using simulated annealing

(3) During testing, a signature is extracted from the DUT by applying the compact image dataset to the DUT. The signature is passed to an outlier detector which predicts whether the DUT is “pass” ($A > A_{cutoff}$) or “fail” ($A \leq A_{cutoff}$).

Steps (1) and (2) are executed offline before real time testing and the outlier detector used in step (3) is trained using the D labeled DUTs provided by the end user.

III. RRAM Variability Model

DNNs consist of convolution, dense, maxpool and batch-normalization layers. Convolution and dense layers involve matrix vector multiplication and general matrix multiplication, which can be accelerated using RRAM crossbars. Within a crossbar, the (p, q)-th element of a weight matrix $W \in \mathbb{R}^{P \times Q}$, w_{pq} is mapped to the (p, q)-th RRAM cell and its conductance is programmed to g_{pq}. We refer the reader to [10] for details about how DNNs are accelerated using analog crossbars.

Accurate inference of RRAM based DNNs relies on the assumption that a DNN weight can be precisely mapped to the target conductance within the crossbar. However, measurements from manufactured RRAM crossbars show that programmed conductance within a crossbar deviates from its target conductance due to device-to-device and cycle-to-cycle variations [11]. Our goal is to model the forward pass of a DNN in presence of RRAM conductance variations.

For variability modeling, we replace the ideal weight matrix of each layer of the DNN W with an equivalent non-ideal weight matrix W^{ni}, which is calculated as: (1) We first flatten the weight matrix $W \in \mathbb{R}^{P \times Q}$ into a flattened weight vector $\overrightarrow{W}^{fl} \in \mathbb{R}^{PQ}$. (2) We calculate a non-ideal flattened weight vector as: $\overrightarrow{W}^{fl,ni} = \overrightarrow{W}^{fl}(\mathbf{1} + \overrightarrow{\epsilon})$ Here, the non-ideality coefficient vector $\overrightarrow{\epsilon} \in \mathbb{R}^{PQ}$ models the impact of conductance variations on the weights of a DNN. (3) Finally, we convert the non-ideal flattened weight vector into a non-ideal weight matrix $W^{ni} \in \mathbb{R}^{P \times Q}$. The weight matrix is flattened into a vector for spatial correlation modeling as discussed in the next paragraph.

Now we explain, how $\overrightarrow{\epsilon}$ is calculated. Prior work has shown that device-to-device variation can be inter-chip or intra-chip [12]. Inter-chip variation (also called systematic variation) affects all RRAM cells of a crossbar equally whereas intra-chip variation affects RRAM cells differently. Further, the research of [13] has shown that RRAM cells which are physically close to each other have correlated intra-chip variation profile. As a result intra-chip variation can be modeled as a sum of spatially correlated variation and independent random variation [13]. Cycle-to-cycle variation can be modeled as independent random variation. Based on these observations, we model the non-ideality coefficient vector as a sum of systematic ($\overrightarrow{\epsilon}^{sys}$), spatially correlated ($\overrightarrow{\epsilon}^{cor}$) and random ($\overrightarrow{\epsilon}^{rand}$) non-ideality coefficient vectors, i.e.,

$$\overrightarrow{\epsilon} = \overrightarrow{\epsilon}^{sys} + \overrightarrow{\epsilon}^{cor} + \overrightarrow{\epsilon}^{rand} \qquad (1)$$

By definition, systematic variation affects all RRAM cells of a DNN equally [12]. As a result, we sample $\psi \sim \mathcal{N}(0, \sigma^2_{sys})$ and set $\overrightarrow{\epsilon}^{sys}_a = \psi$ for $1 \leq a \leq PQ$. Following [14], we sample all elements of $\overrightarrow{\epsilon}^{rand}$ independently from $\mathcal{N}(0, \sigma^2_{rand})$. Here σ^2_{sys} (σ^2_{rand}) refers to the variance in the elements of non-ideality coefficient vector caused by systematic (random) variation. Calculation of $\overrightarrow{\epsilon}^{cor}$ involves the following steps: (1) We compute the spatial correlation matrix $\Gamma \in \mathbb{R}^{PQ \times PQ}$, where the (a, b)-th element of Γ, γ_{ab} is the correlation between a-th and b-th element of $\overrightarrow{W}^{fl}$. Within a crossbar, if w^{fl}_a and w^{fl}_b are d_{ab} distance apart, then we set $\gamma_{ab} = \exp(-\lambda d_{ab})$, where λ is a proportionality constant [15]. (2) The spatial covariance matrix is calculated as $\Sigma = \Gamma \odot \sigma^2_{cor}$, where σ^2_{cor} represents the variance corresponding to spatial correlation. (3) Finally, we sample $\overrightarrow{\epsilon}^{cor}$ from Σ.

We refer the reader to [11]–[15] for more details about RRAM variability modeling.

IV. Image Synthesis

The goal of the synthetic image dataset (SID) generation is to synthesize a set of images for post-manufacture test without any access to training and testing dataset of the DNN. The key intuition is that when a DNN is used to classify the images from its testing dataset, the DNN outputs produce a constant entropy independent of the individual images. The proposed

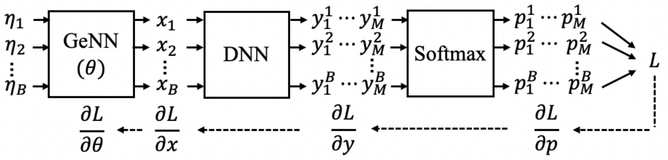

Fig. 2: Generator neural network training

entropy guided image synthesis (EGIS) approach trains a generator neural network (GeNN) which maps a set of input noise vectors to images such that when the synthetic images are applied to the DNN, the DNN outputs produce an entropy equal to the entropy corresponding to the original testing dataset. In Section IV-A, we first introduce definitions of class entropy and sample entropy. *We show that sample entropy is approximately zero and class entropy scales logarithmically with respect to the number of classes in the testing dataset, irrespective of the pixel values of individual images.* Next, in Section IV-B, we explain how the GeNN is trained to create the SID, which has similar class and sample entropies compared to the original testing dataset.

A. Sample and Class Entropy

Assume that an ideal DNN (used as a classifier) is trained for an image dataset $\mathcal{T} = \{x_n, lab_n\}_{n=1}^{N}$, with N images from total M classes. The true label of the n-th image x_n is lab_n. When we apply x_n to the ideal DNN, the DNN's final layer neuron outputs consist of M real numbers $\{y_m^n\}_{m=1}^{M}$. These DNN outputs undergo softmax operation to generate normalized class probabilities as $\{p_m^n\}_{m=1}^{M}$, where p_m^n is the probability that the image belongs to class m. The softmax operation is defined as, $p_m^n = \frac{\exp(y_m^n)}{\sum_{i=1}^{M}\exp(y_i^n)}$. By definition of an ideal DNN (used as a classifier):

$$p_m^n = \begin{cases} 1 & \text{if } m = lab_n \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

In the rest of this subsection, we first introduce the concept of entropy in machine learning. Next, we will use Equation (2) to derive class entropy and sample entropy. Intuitively, class entropy is a measure of the confidence with which a DNN classifies a set of images. A *lower class entropy implies that the images are classified with high confidence*. On the other hand, a *high value of sample entropy indicates that the dataset has a diverse set of images.*

In machine learning, if an image x_n is applied to a DNN for classification and the normalized class probabilities (softmax outputs) are $\{p_m^n\}_{m=1}^{M}$, then the entropy is defined as $-\sum_{m=1}^{M} p_m^n \log p_m^n$. Using this, we define class entropy H_c and sample entropy H_s as follows:
Class Entropy: We derive class entropy as:

$$H_c = -\frac{1}{N}\sum_{n=1}^{N}\sum_{m=1}^{M} p_m^n \log(p_m^n) \tag{3}$$

$$= -\frac{1}{N}\sum_{n=1}^{N}[\sum_{m=lab_n} p_m^n \log(p_m^n) + \sum_{m\neq lab_n} p_m^n \log(p_m^n)]$$

$$= 0 \tag{4}$$

In the above derivation, when $m = lab_n$, we have $\log(p_m^n) = log(1) = 0$. When $m \neq lab_n$, we have $p_m^n = 0$.
Sample Entropy: We define sample entropy as:

$$H_s = -\sum_{m=1}^{M}\big(\frac{1}{N}\sum_{n=1}^{N} p_m^n\big)\log\big(\frac{1}{N}\sum_{n=1}^{N} p_m^n\big) \tag{5}$$

If there are N_m images in $\mathcal{T}$ with label m, we can calculate:

$$\overline{p}_m = \frac{1}{N}\sum_{n=1}^{N} p_m^n = \frac{1}{N}[\sum_{m=lab_n} p_m^n + \sum_{m\neq lab_n} p_m^n]$$

$$= \frac{1}{N}[\sum_{m=lab_n} 1 + \sum_{m\neq lab_n} 0] = \frac{N_m}{N}$$

If all classes have equal number of images in $\mathcal{T}$, we have $N_m = \frac{N}{M}$ and $\overline{p}_m = \frac{1}{M}$. Finally, we derive

$$H_s = -\sum_{m=1}^{M}\overline{p}_m \log \overline{p}_m = -\sum_{m=1}^{M}\frac{1}{M}\log\frac{1}{M} = \log M \tag{6}$$

From Equation (6) and (4), we have:

$$\exp(H_s - H_c) = \exp(\log M - 0) = M \tag{7}$$

If $\mathcal{T}$ is applied to a practical DNN (instead of ideal DNN), Equation (2) holds true approximately (instead of exact equality). As a result, Equation (4),(6) and (7) can be modified as:

$$H_c \approx 0 \tag{8}$$

$$H_s \approx \log M \tag{9}$$

$$\exp(H_s - H_c) \approx M \tag{10}$$

B. Generator Neural Network Training

We want to create a SID $\mathcal{X} = \{x_1, x_2 \cdots, x_N\}$ such that when images from $\mathcal{X}$ are applied to a DNN with nominal weights (which are mapped to RRAM crossbars), it satisfies Equation (8),(9) and (10). To achieve this, we use a GeNN with weights θ, which takes a Ω dimensional noise $\eta \in \mathbb{R}^{\Omega}$ as its input and maps the noise to an image $g_\theta(\eta)$. The weights of the GeNN (θ) are trained using backpropagation as shown in Fig. 2.

Before training, we fix the probability density function of the noise as $\mathbb{P}(\eta) = \mathcal{N}(0, \mathbb{I})$. At every iteration of the optimization, B (B refers to batch size) noise vectors $\{\eta_b\}_{b=1}^{B}$ are sampled from $\mathbb{P}(\eta)$. During forward pass (as shown by solid arrows in Fig. 2), the GeNN generates B images $\{x_b\}_{b=1}^{B}$ corresponding to the noise vectors $\{\eta_b\}_{b=1}^{B}$. The images are then passed to a DNN with trained weights (used in end application). The DNN maps an input image x_b to outputs $\{y_m^b\}_{m=1}^{M}$. The DNN outputs are passed to the softmax function to calculate normalized class probabilities $\{p_m^b\}_{m=1}^{M}$. We calculate class entropy H_c and sample entropy H_s from the softmax outputs.

In order to reduce uncertainty in classification (minimize class entropy) and promote diversity of images (maximize sample entropy), we set the loss as $\mathcal{L} = -\exp(H_s - H_c)$. The goal of GeNN training is to minimize $\mathcal{L}$ by updating θ, i.e., $\min_\theta \mathcal{L}$. During backward pass (as shown by dotted arrows in Fig. 2), the gradient of $\mathcal{L}$ with respect to θ is calculated using chain rule of differentiation. Finally, θ is updated using Adam optimizer.

C. Synthetic Image Dataset

Once the generator neural network is trained, we perform inference using it to create a synthetic image dataset. The synthetic image dataset is created in two steps:
(1) We sample N noise vectors $\{\eta_1, \eta_2, \cdots, \eta_N\}$
(2) Each sampled noise vector is applied to the generator neural network. Corresponding to the input noise η_n, the generator creates a synthetic image x_n (for $1 \leq n \leq N$).

V. Image Set Compaction

The SID is a set of N ($\approx 10^3$) images that can be potentially used for post manufacture test. However, for practical utility during volume production, it is important to test DUTs using as few images as possible. We develop an image set compaction algorithm to create a compact image dataset (CID) of K images ($K << N$), which are used for efficient post-manufacture test. While it is possible to generate only K images using the approach mentioned in Section IV and use them for testing, such an approach leads to unpredictable post-manufacture test performance because the generated images depend highly on the randomly sampled noises for image generation. On the other hand, the image set compaction algorithm uses signatures from DUTs to down-select images from the SID, leading to reliable performance during post-manufacture test.

The image set compaction algorithm requires D DUTs which are known to be either "pass" or "fail". The goal is to choose K images which, when used for predicting the D DUTs as "pass" or "fail", are most effective. In this work, we use simulated annealing to pick the compact image dataset.

Algorithm 1 outlines the overall image set compaction algorithm. As inputs, it requires the SID $\mathcal{X}$, initial and final temperatures T^{init} and T^{final}, and D DUTs which are known to be "pass" or "fail". The algorithm starts with K images randomly selected from $\mathcal{X}$. At every iteration of the optimization:
(1) We replace each image in the current solution with a randomly chosen image from $\mathcal{X}$ with probability 0.5 to generate a new solution. (line 5-12)
(2) To calculate the reward for the new solution, the D DUTs with known label are split into a training set of DUTs D_{tr} and a validation set of DUTs D_{val} (line 16). The ratio of $\frac{|D_{tr}|}{|D_{val}|}$ is set as 4. Each image of the new solution is applied to all the D DUTs to generate a signature (defined in Section VI) corresponding to each DUT (line 17). A binary classifier is trained to predict the label of a DUT from its signature using a training set of DUTs D_{tr} (line 18). The trained classifier then predicts the labels of the validation DUTs D_{val} and A_{val} refers to the percentage of DUTs correctly classified by the binary classifier (line 19-20). The value of A_{val} is calculated for I random trials, where in each trial, training and validation DUTs are picked randomly. The reward for the new solution is set to the average value of A_{val} over I random trials.
(3) If the new reward is better than the current reward, the current solution is updated (line 25-26). Otherwise the new solution is accepted or rejected based on the derived acceptance probability (line 29). The annealing temperature is reduced at the end of each iteration, where ρ is the cooling rate (line 35). At the end, when the temperature reaches below T^{final}, the solution which achieves the best reward is set as the compact image dataset for post-manufacture test.

Algorithm 1 Image Set Compaction

```
 1: Input: Synthetic Image Dataset $\mathcal{X}$, number of images in $\mathcal{X} = N$,
    number of images in compact image dataset $K$, initial temper-
    ature $T^{init}$, final temperature $T^{final}$ and $D$ DUTs which are
    known to be "pass" or "fail"
 2: Initialize Randomly pick an initial solution $\mathcal{C}^{cur} =$
    $\{c_1^{cur}, c_2^{cur}, \cdots c_K^{cur}\} \subset \mathcal{X}$ and set best reward $r^{best} = -\infty$,
    current reward $r^{cur} = -\infty$, temperature $T^{cur} = T^{init}$
 3: while $T^{cur} > T^{final}$ do
 4:   // Step 1: Pick a new solution
 5:   Set $\mathcal{C}^{new} = \mathcal{C}^{cur}$
 6:   for $k = 1$ upto $K$ do
 7:     Sample a random variable $\alpha \sim$ Bernoulli$(0.5)$
 8:     if $\alpha = 1$ then
 9:       Randomly pick $j \in \{1, 2, \cdots, N\}$ such that $x_j \notin \mathcal{C}^{new}$
10:       Set $c_k^{new} = x_j$
11:     end if
12:   end for
13:   // Step 2: Evaluate a reward with new solution
14:   $r^{new} = 0$
15:   for $i = 1$ upto $I$ do
16:     Randomly split $D$ DUTs into $D_{tr}$ and $D_{val}$ DUTs
17:     Generate signatures of all $D$ DUTs by applying $\mathcal{C}^{new}$
18:     Train a binary classifier with $D_{tr}$ DUTs which predicts a
        DUT as "pass" or "fail" from its signature
19:     Predict $D_{val}$ DUTs using the classifier
20:     Set validation accuracy $A_{val} = \frac{\text{\#correct predictions}}{|D_{val}|} \times 100\%$
21:     $r^{new} = r^{new} + A_{val}$
22:   end for
23:   $r^{new} = \frac{r_{new}}{I}$
24:   // Step 3: Accept or reject new solution
25:   if $r^{new} > r^{cur}$ then
26:     $\mathcal{C}^{cur} = \mathcal{C}^{new}$, $r^{cur} = r^{new}$
27:   else
28:     Calculate acceptance probability $p = \exp\frac{(r^{new} - r^{cur})}{T^{cur}}$
29:     Set $\mathcal{C}^{cur}, r^{cur}$ based on the derived probability as
        $$\mathcal{C}^{cur}, r^{cur} = \begin{cases} \mathcal{C}^{cur}, r^{cur} & \text{with probability } 1 - p \\ \mathcal{C}^{new}, r^{new} & \text{with probability } p \end{cases}$$
30:   end if
31:   if $r^{new} > r^{best}$ then
32:     $\mathcal{C}^{best} = \mathcal{C}^{new}$
33:     $r^{best} = r^{new}$
34:   end if
35:   Set $T^{cur} = \rho T^{cur}$
36: end while
37: Return $\mathcal{C}^{best}$
```

VI. Outlier Detection

In this section, we explain the training procedure of the outlier detector (shown in Fig. 1), which is used to label a DUT as "pass" or "fail" from the response of the DUT to the CID. The outlier detector is implemented using a binary classifier. Each image of the CID $\mathcal{C} = \{c_1, c_2 \cdots c_K\}$ is applied to the D DUTs with known "pass"/"fail" labels. When the k-th image in $\mathcal{C}$, c_k is applied to the d-th DUT, the DUT output is a vector of M real numbers $\overrightarrow{s}_k = [y_1^k \quad y_2^k \quad \cdots \quad y_M^k]$. The DUT outputs corresponding to all images in $\mathcal{C}$ are stacked to generate the DUT signature, i.e., $sig_d = [\overrightarrow{s}_1 \quad \overrightarrow{s}_2 \quad \cdots \quad \overrightarrow{s}_K]$.

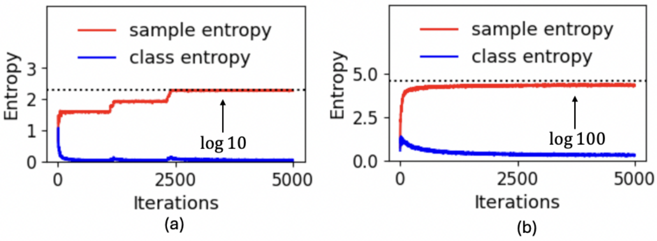

Fig. 3: Sample and class entropy for (a) Mobilenet (CIFAR-10) and (b) VGG16 (CIFAR-100)

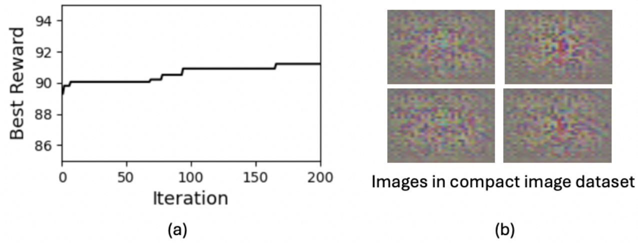

Fig. 4: (a) Best reward in simulated annealing (b) Compact Image Dataset (VGG16, CIFAR-10, $K = 4$)

Each DUT has a label $lab_d \in \{0, 1\}$, where 1 indicates a “pass” DUT and 0 indicates a “fail” DUT. The overall training data for the classifier is D signatures with associated labels. The goal of training is to fit a function $f_{clf}(\cdot) : \mathbb{R}^{KM} \rightarrow \{0, 1\}$ such that $f_{clf}(sig_d) = lab_d$. We use gradient boosting to fit f_{clf}, where an ensemble of weak learners represent f_{clf}. We refer the reader to [16] for further details on gradient boosting.

VII. Results

A. Simulation Setup and Performance Metrics

We evaluate three DNNs: (1) VGG16 [17] trained on CIFAR-10 [18] (nominal accuracy = 93.24%) (2) Mobilenet [19] trained on CIFAR-10 (nominal accuracy = 91.72%) and (3) VGG16 trained on CIFAR-100 (nominal accuracy = 72.36%). For variability modeling, we define total variance $\sigma^2_{tot} = \sigma^2_{sys} + \sigma^2_{cor} + \sigma^2_{rand}$. Following [12], we fix $\sigma^2_{rand} = 0.5 \times \sigma^2_{tot}$. We define percentage of systematic (spatially correlated) variation as $\frac{\sigma^2_{sys}}{\sigma^2_{tot}}$ $(\frac{\sigma^2_{cor}}{\sigma^2_{tot}})$ $\times 100\%$. We simulate for (a) typical variability: 25% (25%) systematic (spatially correlated) variation and (b) extreme systematic variability 50% (0%) systematic (spatially correlated) variation. Table I outlines the layers of the GeNN. We sample each noise vector from $\mathbb{R}^{100}$. After layer 1-4, we use batch normalization and Relu activation. The final layer of the GeNN uses tanh activation. The generator neural network is trained for 5000 iterations using Adam optimizer [20] with an initial learning rate of 0.0003. We use (a) $\lambda = 0.001$ (b) $\sigma_{tot} = 0.3$ for VGG16 and $\sigma_{tot} = 0.15$ for Mobilenet. (c) $T^{init} = 2.0$ (d) $T^{final} = 0.03$ (e) cooling rate $\rho = 0.98$. (f) Number of images in SID $N = 2048$ (g) Number of images in CID $K = 2 - 16$.

We use 1000 DUTS with known labels (“pass” or “fail”) to train the outlier detector. We evaluate the test framework on another 1000 DUTs. During testing, a DUT can fall in either of the four cases: (a) True positive (TP): “pass” predicted as “pass” (b) False negative (FN): “pass” predicted as “fail” (c) False positive (FP): “fail” predicted as “pass” (d) True negative (TN): “fail” predicted as “fail”. We define test accuracy $TA = \frac{TP+TN}{TP+TN+FP+FN} \times 100\%$, test escape as $TE = \frac{FP}{TP+TN+FP+FN} \times 100\%$ and yield loss as $YL = \frac{FN}{TP+TN+FP+FN} \times 100\%$. We categorize test escapes into critical test escapes (CTE) and benign test escapes (BTE). CTE is calculated as the percentage of DUTs which are predicted as “pass” but their classification accuracy lies in the range $A < A_{cutoff} - 1\%$. BTE is calculated as the percentage of DUTs which are predicted as “pass” but their classification accuracy is in the range $A_{cutoff} - 1\% \leq A \leq A_{cutoff}\%$. We define A_p as the p-th percentile classification accuracy of the 1000 DUTs used for training the outlier detector. We evaluate our test framework for $A_{cutoff} = A_{50}, A_{60}$ and A_{70}. For example, for $A_{cutoff} = A_{70}$, 70% DUTs with highest accuracy are referred as “pass”.

TABLE I: Architecture of generator neural network

Layer	Type	Input Channels	Output Channels	Kernel Size
1	Transpose Convolution	100	256	2
2	Transpose Convolution	256	128	4
3	Transpose Convolution	128	64	4
4	Transpose Convolution	64	32	4
5	Transpose Convolution	32	32	4

B. Compact Image Dataset Generation

Fig. 3 shows the class and sample entropy during the training of the GeNN for Mobilenet (CIFAR-10) and VGG16 (CIFAR-100). The class entropy approaches 0 whereas sample entropy approaches $\log M$ ($M = 10$ for CIFAR-10 and $M = 100$ for CIFAR-100) at the end of the training. Fig. 4a shows how the best reward varies over the iterations of simulated annealing (VGG16, CIFAR-10). The best reward, starts at 89 and reaches 91 at the end of the optimization. Fig. 4b shows the final images in the compact image dataset (VGG16, CIFAR-10, $K = 4$).

C. Evaluation of the Test Framework

Fig. 5a shows the performance of the proposed test framework for typical variation (VGG16, CIFAR-10). For $A_{cutoff} = A_{50}$, we achieve test accuracy between 88.6-89.9% depending on the number of images K in the CID. For $A_{cutoff} = A_{60}$ and A_{70}, the test accuracy lies between 90.9-93% and 91.8-93.1% respectively. For all simulation conditions, critical test escape is less than 4.4%, yield loss is less than 4.5% and benign test escape is less than 3.5%. As shown in Fig. 5b and Fig. 5c, the proposed framework achieves similar test accuracy for Mobilenet trained on CIFAR-10 and VGG-16 trained on CIFAR-100. For extreme variability, (50% systematic, 0% spatially correlated and 50% random) the test framework achieves test accuracy in the range 95-98%, critical and benign test escapes less than 2% and yield loss less than 2.4% (VGG16, CIFAR-10). For Mobilenet trained on CIFAR-10 and VGG-16 trained on CIFAR-100, we observe similar trends.

The key takeaways from these experiments are: (a) Test accuracy is above 89% for all simulation conditions. (b) Depending on simulation conditions, 4-8 images are appropriate

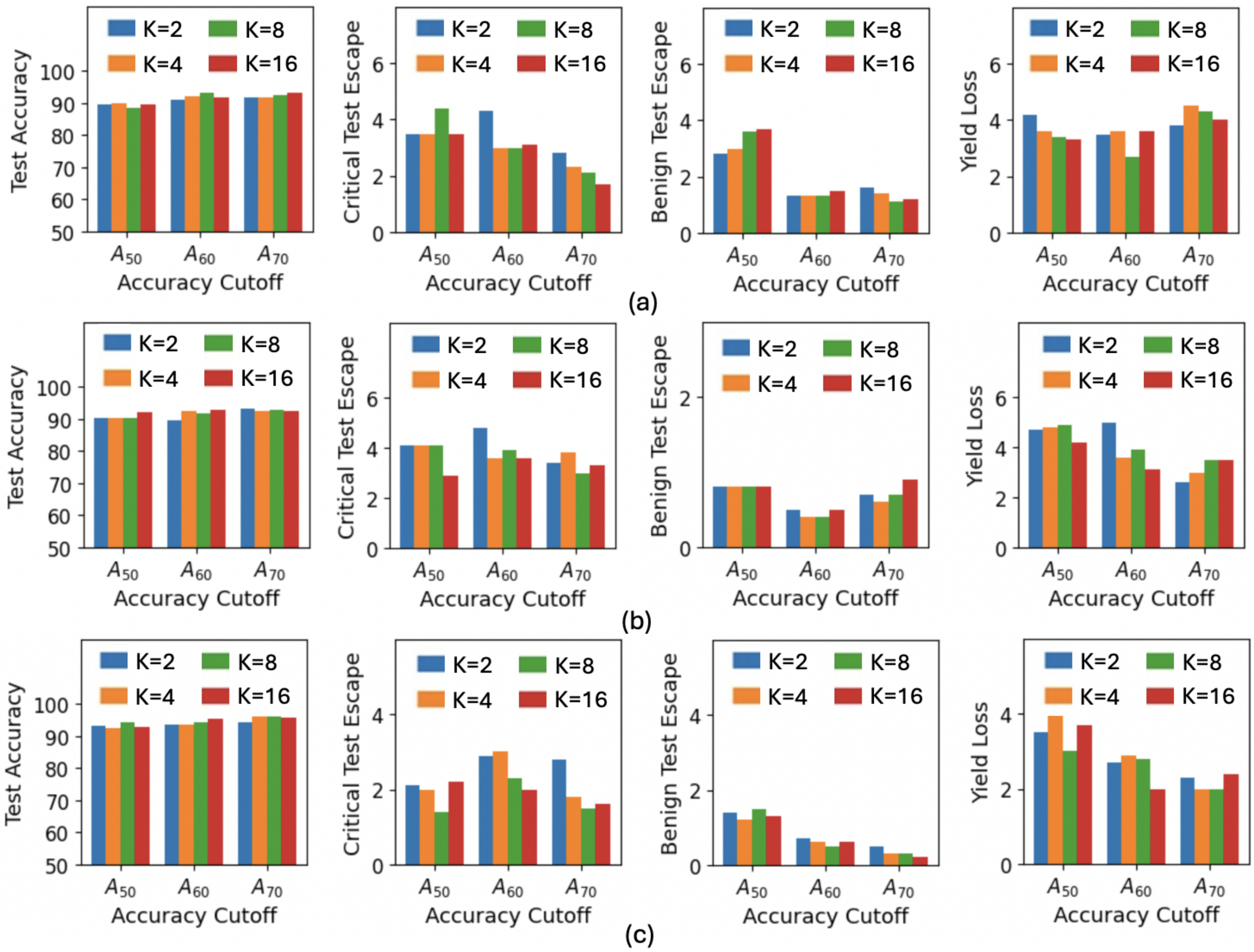

Fig. 5: Evaluation of the test framework for (a) VGG16 on CIFAR-10 (b) Mobilenet on CIFAR-10 (c) VGG16 on CIFAR-100, with 25% systematic, 25% spatially correlated and 50% random variation (K = number of images in the compact image dataset)

TABLE II: Comparison with dataset-aware test

			Test Accuracy	
systematic	correlated	A_{cutoff}	ours	[6]
25%	25%	A_{50}	89.9	87.7
		A_{60}	92.1	93.3
		A_{70}	91.8	93.9
50%	0%	A_{50}	95.4	96.4
		A_{60}	96.4	97.4
		A_{70}	97.8	98.1

for testing. Using 16 images leads to highest test duration whereas using 2 images leads to sub-optimal test accuracy.

D. Runtime Analysis and Comparison with State-of-the-art

GeNN training, image set compaction and outlier detection take 94 seconds, 688 seconds and 9ms respectively using NVIDIA A100 GPU (VGG16, CIFAR-10, $N = 2048$ and $K = 4$). Since the CID is created offline and real-time testing of RRAM-based DNNs only involves outlier detection, the proposed test framework requires 9ms to test a DUT.

We compare the proposed framework with the state-of-the-art dataset aware test of [6] (VGG16, CIFAR-10, $K = 4$). Two major differences between the proposed framework and [6] are: the work of [6] assumes access to (a) all images in the original testing dataset and (b) exact classification accuracy of all 1000 DUTs used to train the outlier detector. The proposed framework only assumes access to "pass"/"fail" labels of 1000 DUTs. Table II shows that despite lack of access to the original testing dataset and exact accuracies of the 1000 DUTs, for typical process variation the proposed dataset-agnostic test achieves test accuracy within 2.1% of [6] for a wide range of accuracy cutoff. For extreme systematic variation the test accuracy of dataset-agnostic test is within 1% of [6]

VIII. Conclusion

We propose a dataset-agnostic post-manufacture test framework for RRAM based DNNs, targeted for privacy critical applications. A novel methodology is proposed to train a generator neural network for creating a synthetic image dataset. An image set compaction algorithm is developed to create a compact image dataset, reducing the test duration during volume production. The proposed framework achieves comparable test accuracy with respect to dataset-aware test.

Acknowledgment

This research was supported by the U.S. National Science Foundation under Grant: 2128149.

References

[1] S. Yu, H. Jiang, S. Huang, X. Peng, and A. Lu, "Compute-in-memory chips for deep learning: Recent trends and prospects," *IEEE Circuits and Systems Magazine*, vol. 21, no. 3, pp. 31–56, 2021.

[2] S. Yu and P.-Y. Chen, "Emerging memory technologies: Recent trends and prospects," *IEEE Solid-State Circuits Magazine*, vol. 8, no. 2, pp. 43–56, 2016.

[3] F. Su, C. Liu, and H.-G. Stratigopoulos, "Testability and dependability of ai hardware: Survey, trends, challenges, and perspectives," *IEEE Design Test*, vol. 40, no. 2, pp. 8–58, 2023.

[4] A. Chaudhuri, C.-Y. Chen, J. Talukdar, and K. Chakrabarty, "Functional test generation for ai accelerators using bayesian optimization," in *2023 IEEE 41st VLSI Test Symposium (VTS)*, 2023, pp. 1–6.

[5] S. A. El-Sayed, T. Spyrou, L. A. Camuñas-Mesa, and H.-G. Stratigopoulos, "Compact functional testing for neuromorphic computing circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 42, no. 7, pp. 2391–2403, 2023.

[6] K. Ma, A. Saha, C. Amarnath, and A. Chatterjee, "Efficient low cost alternative testing of analog crossbar arrays for deep neural networks," in *2022 IEEE International Test Conference (ITC)*, 2022, pp. 499–503.

[7] S. T. Ahmed and M. B. Tahoori, "Compact functional test generation for memristive deep learning implementations using approximate gradient ranking," in *2022 IEEE International Test Conference (ITC)*, 2022, pp. 239–248.

[8] F. Mireshghallah, M. Taram, P. Vepakomma, A. Singh, R. Raskar, and H. Esmaeilzadeh, "Privacy in deep learning: A survey," *CoRR*, vol. abs/2004.12254, 2020. [Online]. Available: https://arxiv.org/abs/2004.12254

[9] S. Kundu, S. Banerjee, A. Raha, F. Su, S. Natarajan, and K. Basu, "Trouble-shooting at gan point: Improving functional safety in deep learning accelerators," *IEEE Transactions on Computers*, vol. 72, no. 8, pp. 2194–2208, 2023.

[10] M. Dazzi, A. Sebastian, L. Benini, and E. Eleftheriou, "Accelerating inference of convolutional neural networks using in-memory computing," *Frontiers in Computational Neuroscience*, vol. 15, 2021.

[11] A. Fantini, L. Goux, R. Degraeve, D. Wouters, N. Raghavan, G. Kar, A. Belmonte, Y.-Y. Chen, B. Govoreanu, and M. Jurczak, "Intrinsic switching variability in hfo2 rram," in *2013 5th IEEE International Memory Workshop*, 2013, pp. 30–33.

[12] Z. Deng and M. Orshansky, "Variability-aware training and self-tuning of highly quantized dnns for analog pim," in *2022 Design, Automation Test in Europe Conference Exhibition (DATE)*, 2022, pp. 712–717.

[13] Y. Zhu, G. L. Zhang, T. Wang, B. Li, Y. Shi, T.-Y. Ho, and U. Schlichtmann, "Statistical training for neuromorphic computing using memristor-based crossbars considering process variations and noise," in *2020 Design, Automation Test in Europe Conference Exhibition (DATE)*, 2020, pp. 1590–1593.

[14] Y. Long, X. She, and S. Mukhopadhyay, "Design of reliable dnn accelerator with un-reliable reram," in *2019 Design, Automation Test in Europe Conference Exhibition (DATE)*, 2019, pp. 1769–1774.

[15] J. Xiong, V. Zolotov, and L. He, "Robust extraction of spatial correlation," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 26, no. 4, pp. 619–631, 2007.

[16] J. H. Friedman, "Greedy function approximation: A gradient boosting machine," *The Annals of Statistics*, vol. 29, no. 5, pp. 1189–1232, 2001. [Online]. Available: http://www.jstor.org/stable/2699986

[17] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[18] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," 2009.

[19] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[20] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2017.