Efficient Optimized Testing of Resistive RAM Based Convolutional Neural Networks

Anurup Saha, Kwondo Ma, Chandramouli Amarnath and Abhijit Chatterjee
School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta GA 30332
Email: asaha74@gatech.edu, kma64@gatech.edu, chandamarnath@gatech.edu, abhijit.chatterjee@ece.gatech.edu

2024 IEEE 30th International Symposium on On-Line Testing and Robust System Design (IOLTS) | 979-8-3503-7055-3/24/$31.00 ©2024 IEEE | DOI: 10.1109/IOLTS60994.2024.10616092

***Abstract*—Resistive random access memory (RRAM) based memristive crossbar arrays enable low power and low latency inference for convolutional neural networks (CNNs), making them suitable for deployment in IoT and edge devices. However, RRAM cells within a crossbar suffer from conductance variations, making RRAM-based CNNs vulnerable to degradation of their classification accuracy. To address this, the classification accuracy of RRAM based CNN chips can be estimated using predictive tests, where a trained regressor predicts the accuracy of a CNN chip from the CNN's response to a compact test dataset. In this research, we present a framework for *co-optimizing the pixels of the compact test dataset and the regressor*. The novelty of the proposed approach lies in the ability to co-optimize individual image pixels, overcoming barriers posed by the computational complexity of optimizing the large numbers of pixels in an image using state-of-the-art techniques. The co-optimization problem is solved using a three step process: a greedy image down-selection followed by backpropagation driven image optimization and regressor fine-tuning. Experiments show that the proposed test approach reduces the CNN classification accuracy prediction error by 31% compared to the state of the art. It is seen that a compact test dataset with only 2-4 images is needed for testing, making the scheme suitable for built-in test applications.**

***Index Terms*—Resistive Random-Access Memory (RRAM), Convolutional neural network (CNN) testing, Predictive testing**

I. Introduction

Convolutional neural networks (CNN) are widely used in computer vision because of the shift invariance property of convolution [1]. As a consequence, CNNs are being increasingly deployed in resource-constrained edge devices. For accelerating CNNs, traditional Von-Neumann architectures suffer from the memory wall problem where the physical separation of processing elements and main memory limits energy efficiency and latency. To alleviate this, compute-in-memory has been proposed as a new computing paradigm [2]. Resistive random access memory (RRAM) is a promising technology for implementing compute-in-memory because of its non-volatile storage, moderately high switching speed and high endurance [3]. Despite low inference energy and latency of RRAM based CNNs [4], device-to-device and cycle-to-cycle conductance variations of RRAM cells within a crossbar impact performance (measured as classification accuracy) of RRAM based CNNs [5]. As a result, RRAM based CNNs need to be performance-screened before being deployed in safety critical applications.

Functional tests for RRAM based CNNs have been proposed [6], [7], with a key focus on minimizing the number of images used into a *compact test image subset* (e.g. as opposed to all 10,000 images in the CIFAR-10 dataset). The work of [6] clusters the images from the original test dataset and chooses one image from each cluster to test DNNs. In [8], sensitivity analysis is performed on all original test dataset images and the images with highest sensitivity scores are picked for functional test. The work of [9] uses generative adversarial networks (GANs) to generate test images that improve detection coverage of hard faults in DNN accelerators. Broadly, in the above, the compact test image set used is picked to be a subset of a predefined set of images. Inspired from alternate testing of analog circuits [10], predicting classification accuracy of RRAM-based CNNs from its response to a compact test using a regressor is proposed in [6].

Over and above the test image selection methods of prior research, we argue that *pixel-level optimization of test images* with an appropriate loss metric is necessary to *maximize the precision with which the classification accuracy of a CNN can be predicted* from its test response signature. This has the additional benefit that the number of test images in the compact test dataset can be reduced to 2-4 from the 10-30 images of [6]. This further impacts the efficiency of manufacturing test while enabling built-in testing of RRAM-based CNNs with significantly reduced test latency. In this work, we leverage backpropagation-driven pixel level image optimization to build a compact test dataset that permits precise prediction of CNN classification accuracy from its observed test response. We first formulate an optimization objective that requires co-optimization of the compact test dataset and the regressor used for performance prediction. Next, we minimize the optimization objective using a three step process: (a) greedy image down-selection (b) image optimization, which is initialized with the down-selected images and (c) regressor finetuning. In summary, the key contributions of this work are:

(a) The *first pixel-level backpropagation guided image optimization algorithm to generate the compact test dataset for such predictive tests.* The proposed algorithm outperforms state of the art compact test generation algorithms in terms of prediction error and device misclassification rate.

(b) *A mathematical framework* for co-optimizing the compact test dataset and regressor for designing performance predictive tests for RRAM based CNNs.

(c) *Experimental validation* showing that the classification accuracy of RRAM based CNNs can be predicted using a compact test dataset of only 2-4 images. This makes the proposed test framework suitable for built-in test applications.

979-8-3503-7055-3/24/$31.00 ©2024 IEEE

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on September 15,2024 at 16:58:05 UTC from IEEE Xplore. Restrictions apply.

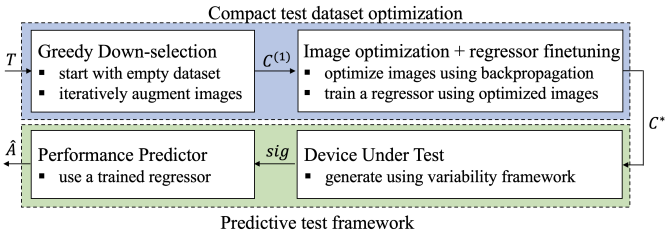


Figure 1: Overall testing approach

II. Approach Overview

A. Predictive Test Framework

In this work, we adopt the predictive testing framework for CNNs, proposed in [6]. In this approach, a compact test dataset is applied to a device under test (DUT) for performance prediction. The performance of RRAM-based CNN (also called DUT) refers to its classification accuracy. We assume that (a) the d-th DUT maps an input image $z \in \mathbb{R}^{Im_c \times Im_x \times Im_y}$ to output $\theta_d(z) \in \mathbb{R}^K$. Here Im_c, Im_x and Im_y refer to the number of channels, rows and columns of an input image, K refers to the number of classes in the training dataset of the CNN and $\theta_d(z)$ refers to the output of the last dense layer of the CNN. (b) Compact test dataset $\mathcal{C}$ consists N_c images, i.e., $\mathcal{C} \in \mathbb{R}^{N_c \times (Im_c \times Im_x \times Im_y)}$. We define test response of a DUT under this compact test dataset as $TR = \theta_d(\mathcal{C})$, where $TR \in \mathbb{R}^{N_c \times K}$. We flatten the test response matrix into a vector to generate the *DUT signature*. Mathematically, we denote the flatten operation as $fl(z)$ where $z \in \mathbb{R}^{M \times N}$ and $fl(z) \in \mathbb{R}^{MN}$. The DUT signature can be defined as $sig_d = fl(\theta_d(\mathcal{C}))$. The DUT signature is passed to a trained regressor for performance prediction. Assuming that the regressor maps an input $z \in \mathbb{R}^{N_c \cdot K}$ to $\theta_{reg}(z) \in \mathbb{R}$, the regressor predicts the accuracy of the DUT as $\hat{A}_d = \theta_{reg}(fl(\theta_d(\mathcal{C})))$. The predictive test can be used for post-manufacture test as well as online built-in test (explained in Section V-C).

B. Optimization Objective

A performance predictive test should predict the accuracy of a DUT $\hat{A}_d$ equal to its actual classification accuracy, i.e., $\hat{A}_d = A_d$. We assume that under the impact of process variation, the outputs of RRAM based CNNs $\theta_d(z)$ are distributed according a probability density function $\mathbb{P}(\theta_d(z))$. The key intuition is that $\hat{A}_d = A_d$ should hold true for any given DUT. As a result, we define a cost function,

$$\mathcal{J} = \mathbb{E}_{\theta_d(z) \sim \mathbb{P}(\theta_d(z))} \left(\theta_{reg}(fl(\theta_d(\mathcal{C}))) - A_d \right)^2 \quad (1)$$

Enhancing the predictive test performance requires minimizing the cost function with respect to θ_{reg} and $\mathcal{C}$. Hence, the overall objective is to find:

$$\theta^*_{reg},\ \mathcal{C}^* = \arg\min_{\theta_{reg}, \mathcal{C}} \mathcal{J} \quad (2)$$

Here $\mathcal{C}^*$ refers to the optimized compact test dataset and θ^*_{reg} refers to the optimized regressor parameters. This research solves the optimization problem of Equation (2) in the following three steps:

① *Greedy image down-selection*: We first impose a constraint that the compact test dataset ($\mathcal{C}$) is a subset of the original test dataset ($\mathcal{T}$), i.e., $\mathcal{C} \subset \mathcal{T}$, and find an approximate solution of Equation (2) using a greedy algorithm (explained in Section IV).

② *Image optimization*: We optimize each pixel value of the down-selected images using gradients of the cost function $\mathcal{J}$ with respect to the pixels of the images in the compact test dataset. The optimized images at the end of this step constitute the optimized compact test dataset for predictive test (explained in Section V-A).

③ *Regressor fine-tuning*: Finally, we fix the optimized compact test dataset as $\mathcal{C}^*$ and train a regressor with parameters θ^*_{reg} for DUT performance prediction (explained in Section V-B).

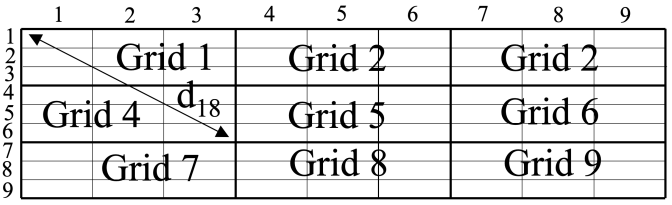


Figure 2: Split a crossbar for spatially correlated variability modeling (d_{18} is the distance between grid 1 and 8)

III. Preliminaries

A. RRAM Crossbar

To accelerate vector matrix multiplication (VMM), a major computational kernel of CNNs, RRAM crossbars are used, where each weight w_{ij} of a CNN model is stored using an RRAM cell of equivalent conductance g_{ij}. Each RRAM device within a crossbar operates according to Ohm's law, where the output current is proportional to the voltage applied across its terminals. Digital to analog converters provide input voltages to the crossbar along the rows. VMM output is available in the form of currents across crossbar columns and the output current is converted to real numbers using analog to digital converters. We refer the reader to [11] for further details on how an RRAM crossbar accelerates CNN inference.

B. Variability Injection Framework

Conductance measurements from fabricated RRAM crossbars show that conductance of RRAM cells within a crossbar deviates from target conductance due to device-to-device and cycle-to-cycle variations [12]. The goal is to quantify the impact of RRAM conductance variation on the VMM outputs on a layer by layer basis by replacing the ideal weight matrix (W^l) of every CNN layer with a non-ideal weight matrix (W^l_{ni}). While RRAM crossbars are vulnerable to stuck-at-faults as well, the impact of stuck-at-faults on inference accuracy of RRAM-based CNNs can be completely recovered using fault aware remapping [13]. As a result, in this work we focus on conductance variation only. Since RRAM cells within a crossbar are laid out in a rectangular manner, the variability injection framework assumes input weights to be two-dimensional matrices. We convert convolution layer weights of shape $C_{out} \times C_{in} \times X \times Y$ into two dimensional matrices of shape $C_{out} \times C_{in}XY$ before passing it to the variability injection framework. Here C_{out} refers to the number of filters

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on September 15,2024 at 16:58:05 UTC from IEEE Xplore. Restrictions apply.

in a convolution layer and C_{in}, X and Y refer to the number of channels, rows and columns of the convolution filters.

Algorithm 1 Variability Injection Framework

1: **Input**: Weights of a pretrained CNN model: $W^{CNN} = \{W^1, W^2, \cdots, W^L\}$
2: **Goal**: Calculate non-ideal CNN weights accounting effects of process variation $W_{ni}^{CNN} = \{W_{ni}^1, W_{ni}^2, \cdots, W_{ni}^L\}$
3: Sample $\epsilon^{sys} \sim \mathcal{N}(0, \sigma_{sys}^2)$
4: **for** $l = 1$ upto L **do**
5: Assume $W^l \in \mathbb{R}^{M \times N}$ and initialize $\mathcal{E}^{sys}, \mathcal{E}^{rnd}$ and $\mathcal{E}^{cor} \in \mathbb{R}^{M \times N}$ with all 0s
6: **for** $m \in \{1, 2, \cdots, M\}$ **do**
7: **for** $n \in \{1, 2, \cdots, N\}$ **do**
8: $\mathcal{E}^{sys}[m, n] = \epsilon^{sys}$
9: $\mathcal{E}^{rnd}[m, n] \sim \mathcal{N}(0, \sigma_{rnd}^2)$
10: **end for**
11: **end for**
12: Assume that number of rows and columns within a grid are X_1 and X_2 respectively
13: Number of grids along rows $G = \frac{M}{X_1}$
14: Number of grids along columns $H = \frac{N}{X_2}$
15: **for** $i \in \{1, 2, \cdots, GH\}$ **do**
16: **for** $j \in \{1, 2, \cdots, GH\}$ **do**
17: Calculate distance between i-th and j-th grids d_{ij}
18: Set $\Omega[i, j] = \exp(-\lambda d_{ij})$
19: **end for**
20: **end for**
21: Set $\Sigma = \sigma_{cor}^2 \cdot \Omega$
22: Factorize $\Sigma = AA^T$, where A is a lower triangular matrix
23: Sample $\overrightarrow{\eta} \in \mathbb{R}^{GH}$, such that $\overrightarrow{\eta} \sim \mathcal{N}(0, \mathbb{I})$
24: Set $\overrightarrow{\Psi} = A\overrightarrow{\eta}$
25: **for** $m \in \{1, 2, \cdots, M\}$ **do**
26: **for** $n \in \{1, 2, \cdots, N\}$ **do**
27: Set $\mathcal{E}^{cor}[m, n] = \overrightarrow{\Psi}[(\lceil \frac{m}{X_1} \rceil - 1)H + \lceil \frac{n}{X_2} \rceil]$
28: **end for**
29: **end for**
30: Set $W_{ni}^l = W^l + W^l \odot (\mathcal{E}^{sys} + \mathcal{E}^{rnd} + \mathcal{E}^{cor})$
31: **end for**

Prior work on process variation modeling has shown that, device-to-device variation is a combination of intra-chip and inter-chip variations [14]. Moreover, RRAM cells, which are physically close, show similar intra-chip variation characteristics [15]. As a result, we decompose intra-chip variation as a combination of spatially correlated variation and random variation. Finally, cycle-to-cycle variation can be modeled as independent random variation. In summary, we model process variation as sum of: (a) inter-chip or systematic variation (b) independent random variation (c) spatially correlated variation.

Algorithm 1 outlines the entire variability injection framework. It starts with the weights of the pre-trained CNN model W^{CNN}, where W^l is the weight matrix corresponding to the l-th layer of the CNN. The end goal is to calculate three matrices of systematic, random and spatially correlated non-ideality factors and calculate a non-ideal weight matrix W_{ni}^l to capture the impact of conductance variations on VMM outputs. By definition, inter-chip variation affects all weights of a DUT equally. Following [16], we sample only one systematic non-ideality factor from a zero mean normal distribution with variance σ_{sys}^2 (line 3). We sample each element of $\mathcal{E}^{rnd}$ from another zero mean normal distribution with variance σ_{rnd}^2 (line 9). Finally, to sample spatially correlated non-ideality factors, we split an RRAM crossbar into grids (line 12-14). Fig. 2 shows a 9×9 crossbar ($M = N = 9$), which is split into total 9 grids ($G = H = 3$). Each grid has 3 rows and 3 columns of RRAM cells ($X_1 = X_2 = 3$). To calculate spatially correlated non-ideality factors, first spatial correlation between i-th and j-th grids are calculated, where the grid distance between the two grids are d_{ij} (line 18). For example, in Fig. 2, grid 1 and grid 8 are 6 rows and 3 columns apart. As a result $d_{18} = \sqrt{(6^2 + 3^2)} = 3\sqrt{5}$. The corresponding covariance matrix is calculated in line 21, where σ_{cor}^2 refers to the variance corresponding to the spatially correlated variation. Our goal is to sample spatially correlated non-ideality factors from this covariance matrix. We factorize Σ using Cholesky decomposition and a vector of GH standard normal variables are sampled (line 22-23). Finally, one spatially correlated non-ideality factor is calculated corresponding to each grid of the crossbar (line 24). The spatially correlated non-ideality factor of each RRAM cell is set to the spatially correlated non-ideality factor of the grid it belongs to (line 27). Finally, the desired non-ideal weight matrix W_{ni}^l is derived from the ideal matrix W and the matrices of non-ideality factors (line 30).

IV. Greedy Image Down-selection

Algorithm 2 Greedy Image Down-Selection

1: Fix N_c (number of images in compact test dataset) and N_1 (number of random trials) and sample D DUTs (d-th DUT maps input z to output $\theta_d(z)$) with known accuracy A_d for $1 \leq d \leq D$
2: $\mathcal{C} = \{\}$
3: **for** $ni = 1$ upto N_c **do**
4: set $err_{best} = \infty$
5: **for** $i = 1$ upto N_1 **do**
6: Pick a random image $im_i \in \mathcal{T} - \mathcal{C}$
7: Set $C_{cur} = C \cup im_i$
8: Evaluate signature $sig_d = fl(\theta_d(C_{cur}))$ for $1 \leq d \leq D$
9: Fit a regressor (θ_{reg}) which maps sig_d to A_d, i.e., $\theta_{reg}(sig_d) \approx A_d$
10: Evaluate error metric $err_i = \frac{1}{D} \sum_{d=1}^{D} [\theta_{reg}(sig_d) - A_d]^2$
11: **if** $err_i < err_{best}$ **then**
12: $err_{best} = err_i$
13: $C_{best} = C_{cur}$
14: $\theta_{reg}^{best} = \theta_{reg}$
15: **end if**
16: **end for**
17: Set $C = C_{best}$
18: Set $\theta_{reg} = \theta_{reg}^{best}$
19: **end for**
20: return θ_{reg}, $\mathcal{C}$

The purpose of the greedy image down-selection is to provide a good initial guess to the image optimization algorithm. We choose N_c images from the original test dataset $\mathcal{T}$, which provide best prediction results. Mathematically, we can write the objective as:

$$\theta_{reg}^1, \mathcal{C}^1 = \arg \min_{\theta_{reg}, \mathcal{C} \subset \mathcal{T}} \mathbb{E}_{\theta_d(z) \sim \mathbb{P}(\theta_d(z))} \left(\theta_{reg}(fl(\theta_d(\mathcal{C}))) - A_d \right)^2 \quad (3)$$

It should be noted that Equation (2) represents an unconstrained optimization for $\mathcal{C}$, whereas Equation (3) represents a constrained optimization with $\mathcal{C} \subset \mathcal{T}$. Moreover, $\mathbb{P}(\theta_d(z))$ does not have a closed form expression. We can only sample

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on September 15,2024 at 16:58:05 UTC from IEEE Xplore. Restrictions apply.

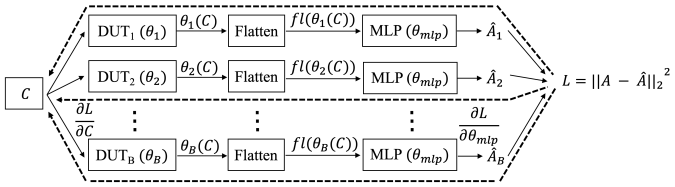


Figure 3: Image optimization using backpropagation: solid line shows forward pass, dashed line shows backward pass

DUTs from the variability injection framework and observe $\theta_d(z)$ from the outputs of sampled DUTs. As a result, we sample D DUTs from the variability injection framework and modify the optimization objective as:

$$\theta^1_{reg}, \mathcal{C}^1 = \arg\min_{\theta_{reg}, \mathcal{C} \subset \mathcal{T}} \frac{1}{D} \sum_{d=1}^{D} \left(\theta_{reg}(fl(\theta_d(\mathcal{C}))) - A_d \right)^2 \quad (4)$$

The problem of creating $\mathcal{C}^1$ with total N_c images is a combinatorial optimization problem with search space size = $\binom{|\mathcal{T}|}{N_c} \approx |\mathcal{T}|^{N_c}$. Here $|\mathcal{T}|$ refers to the number of images in the original test dataset $\mathcal{T}$. We propose a greedy algorithm to approximately solve this search problem in linear time as explained in Algorithm 2. The algorithm starts with an empty set $\mathcal{C}$ (line 2) and after every iteration of the optimization, adds one image to the existing compact test dataset. Within every iteration of the optimization, the greedy algorithm augments a randomly picked image from $\mathcal{T} - \mathcal{C}$ to generate the current test stimulus $\mathcal{C}_{cur}$ (line 6-7). Next, $\mathcal{C}_{cur}$ is applied to all D DUTs to generate D signatures (line 8). A regressor is trained to predict the accuracy of a DUT from its signature (line 9). Among N_1 randomly picked images, we choose the image with minimum regressor error.

V. Image Optimization and Regressor Fine-tuning

A. Image Optimization

In image down-selection algorithm, we imposed a restriction $\mathcal{C} \subset \mathcal{T}$ to find a good initial solution. In the image optimization step, we use the obtained images to initialize image optimization. The image optimization procedure uses the gradient of the cost function $\mathcal{J}$ with respect to $\mathcal{C}$, which is calculated using using backpropagation. In this step we use a multi layer perceptron (MLP) to predict DUT accuracy from signature because all layers of an MLP are differentiable, enabling backpropagation. Algorithm 3 explains the overall image optimization algorithm.

At the beginning, the compact test dataset $\mathcal{C}$ is carefully initialized, whereas the MLP parameters are randomly initialized. As a result, for first E_1 epochs we only update the MLP parameters. We call this step the MLP initialization step. At every iteration, we randomly pick B DUTs from the initially sampled D DUTs (line 6). We keep $\mathcal{C}$ fixed and compute the DUT signature sig_b and the DUT accuracy $\hat{A}_b$ (line 7-8). We calculate mean squared loss between the predicted accuracy and the actual accuracy of the DUTs (line 11). We calculate gradient of the loss with respect to MLP parameters and update the MLP parameters using Adam optimizer (line 12-13).

After the MLP initialization step, we perform image optimization. At every iteration, mean squared loss is evaluated between actual and predicted accuracy of the DUTs (line 20 - 24). In odd iterations of the optimization, we calculate gradient of the loss with respect to the compact test dataset and update $\mathcal{C}$ using Adam optimizer (line 26-27). On the other hand, in even iterations of the optimization, MLP parameters are updated (line 30-31). Fig. 3 shows the forward pass (shown using solid line) and backward pass (shown using dotted line) of backpropagation. Forward pass involves calculating the loss function whereas backward pass involves propagating gradients of loss using chain rule of differentiation. At the end, the optimized compact test dataset is returned.

Algorithm 3 Image Optimization

1: Initialize compact test dataset $\mathcal{C} = \mathcal{C}^1$ and randomly initialize MLP parameters θ_{mlp}. Sample D DUTs with accuracy A_d
2: Set number of epochs for MLP initialization = E_1 and image optimization = E_2 and number of iterations per epoch = I
3: // Step 1: MLP Initialization
4: **for** epoch $= 1$ upto E_1 **do**
5: **for** iteration = 1 upto I **do**
6: Sample B out of total D DUTs
7: **for** $b = 1$ upto B **do**
8: Generate signature of b-th DUT $sig_b = fl(\theta_b(\mathcal{C}))$
9: Predict accuracy using MLP $\hat{A}_b = \theta_{mlp}(sig_b)$
10: **end for**
11: Calculate loss function $\hat{\mathcal{L}} = \frac{1}{B}\sum_{b=1}^{B}[\hat{A}_b - A_b]^2$
12: Calculate $\frac{\partial \hat{\mathcal{L}}}{\partial \theta_{mlp}}$
13: update $\theta_{mlp} = f_{adam}(\theta_{mlp}, \frac{\partial \hat{\mathcal{L}}}{\partial \theta_{mlp}})$
14: **end for**
15: **end for**
16: //Step 2: Image optimization
17: **for** epoch $= 1$ upto E2 **do**
18: **for** iteration = 1 upto I **do**
19: Sample B out of total D DUTs
20: **for** $b = 1$ upto B **do**
21: Generate signature of b-th DUT $sig_b = fl(\theta_b(\mathcal{C}))$
22: Predict accuracy using MLP $\hat{A}_b = \theta_{mlp}(sig_b)$
23: **end for**
24: Calculate loss function $\hat{\mathcal{L}} = \frac{1}{B}\sum_{b=1}^{B}[\hat{A}_b - A_b]^2$
25: **if** iteration is odd **then**
26: Calculate $\frac{\partial \hat{\mathcal{L}}}{\partial \mathcal{C}}$
27: update $\mathcal{C} = f_{adam}(\mathcal{C}, \frac{\partial \hat{\mathcal{L}}}{\partial \mathcal{C}})$
28: **end if**
29: **if** iteration is even **then**
30: Calculate $\frac{\partial \hat{\mathcal{L}}}{\partial \theta_{mlp}}$
31: update $\theta_{mlp} = f_{adam}(\theta_{mlp}, \frac{\partial \hat{\mathcal{L}}}{\partial \theta_{mlp}})$
32: **end if**
33: **end for**
34: **end for**
35: Return $\mathcal{C}$

B. Regressor Fine-tuning

The image optimization step computes the optimized compact test dataset $\mathcal{C}^*$. To develop a predictive test, a regressor needs to be trained as explained in Section II-A. The objective of the regressor fine-tuning step is to find the optimized regressor parameters θ^*_{reg} and use it for performance prediction. Formally, it can be stated as:

$$\theta^*_{reg} = \arg\min_{\theta_{reg}} \frac{1}{D} \sum_{d=1}^{D} \left(\theta_{reg}(fl(\theta_d(\mathcal{C}^*))) - A_d \right)^2 \quad (5)$$

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on September 15,2024 at 16:58:05 UTC from IEEE Xplore. Restrictions apply.

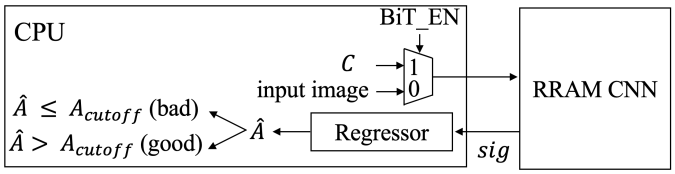


Figure 4: Built-in Test

To find θ^*_{reg}, we apply $\mathcal{C}^*$ to the D DUTs and generate a set of signatures $\{sig_d\}_{d=1}^{D}$. If the accuracy of the d-th DUT is A_d, we fit θ^*_{reg} such that $\theta^*_{reg}(sig_d) \approx A_d$ using gradient boosting regressor [17] from the scikit learn library [18].

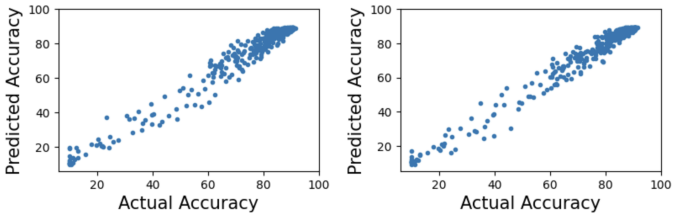
C. Application: Built-in Test

In Section VI, we will show that the classification accuracy of a RRAM CNN can be predicted using a compact test dataset, containing 2-4 images. We propose a built-in test framework for DUT classification, as shown in Fig. 4. In a system on chip (SoC), the compact test dataset is stored in the memory associated with the central processing unit (CPU). During normal operation BiT_EN = 0 (BiT_EN is the enable bit for built-in test mode) and the RRAM CNN performs inference on input images. When the built in test is enabled (BiT_EN = 1), the images of the compact test dataset are applied to the RRAM CNN and CNN sends the signature (defined in Section II) back to the CPU. The computations related to the regressor prediction are performed on the CPU to estimate the DUT accuracy $\hat{A}$. If the application demands that the CNN operate above an accuracy threshold A_{cutoff}, based on the estimated accuracy the RRAM CNN can be classified as "good" ($\hat{A} > A_{cutoff}$) or "bad" ($\hat{A} \leq A_{cutoff}$). Since the built-in test requires applying only 2-4 images to the RRAM CNN, the non-operating cycles of the CNN can be used for applying the test.

VI. Experimental Results

A. Experimental Setup

We evaluate the proposed framework for a VGG-16 CNN [19] trained for CIFAR-10 [20] dataset using PyTorch [21]. Assuming that variances of systematic, spatially correlated and random non-ideality factors are σ^2_{sys}, σ^2_{cor} and σ^2_{rand}, the total variance is calculated as $\sigma^2_{tot} = \sigma^2_{sys} + \sigma^2_{cor} + \sigma^2_{rand}$. We define the proportionality of systematic variation as $\frac{\sigma^2_{sys}}{\sigma^2_{tot}}$ and proportionality of spatially correlated and random variation are calculated similarly. Following [16], we fix random variation as 50% of total variation. We simulate for different proportions of systematic and correlated variations. We use $\lambda = 0.001$, $X_1 = X_2 = 50$ (in Algorithm 1) and $\sigma_{tot} = 0.3$. We first sample $D = 1000$ DUTs and use them for image down-selection, image optimization and regressor fine-tuning. We then independently sample another 500 DUTs which are used for evaluating the proposed compact test dataset optimization framework. We use $N_1 = 20$ (Algorithm 2), $E_1 = 10$, $E_2 = 50$ (Algorithm 3). We use $N_c = 2$ and 4. During image optimization, Adam optimizer [22] is used with initial learning rate of 0.0001. The regressors are trained using gradient boosting regressor from Scikit learn library [18]. We also evaluate the proposed framework for Lenet CNN [23] trained for MNIST dataset [24].



(a) (b)

Figure 5: Actual accuracy vs predicted accuracy of DUTs for VGG-16 with (a) $N_c = 2$ (b) $N_c = 4$

Table I: Simulation conditions

Architecture	Dataset	Baseline accuracy	Avg Accuracy	Accuracy range
VGG-16	CIFAR-10	93.24	77.83	10.0 - 91.6
Lenet	MNIST	99.18	98.23	85.54 - 99.05

For systematic and correlated variability of 25% each, Table I shows the average, minimum and maximum accuracy of 1000 DUTs sampled for compact test dataset generation. It is seen that Lenet CNN trained for MNIST suffers less accuracy degradation due process variation compared to VGG-16 CNN trained for CIFAR-10. Hence, for VGG-16. we evaluate the proposed framework for a cutoff accuracy range of 70% to 85% and for Lenet, we evaluate the proposed framework for a cutoff accuracy range of 95% to 97%.

B. Regressor Performance

If the d-th DUT has accuracy A_d and the regressor predicts its accuracy as $\hat{A}_d$, we define regression error as $e_d = \hat{A}_d - A_d$. We define mean absolute error $MAE = \frac{1}{500}\sum_{d=1}^{500}|e_d|$. We define error standard deviation as $STD = \sqrt{\frac{1}{500}\sum_{d=1}^{500}(e_d - \frac{1}{500}\sum_{i=1}^{500}e_i)^2}$. Ideally, the rergessor should have low MAE and low STD. For VGG-16 CNN and 25% systematic and 25% correlated variability, Fig. 5 shows that the DUT accuracies predicted by the regressor closely tracks the actual accuracies of the DUTs. Table II shows the mean absolute error and error standard deviation of the regressor for different percentages of systematic and spatially correlated variability and for $N_c = 2$ and 4. For all simulation condition, we observe that mean absolute error is within 2.6% and error standard deviation is within 4%. As N_c increases, the length of the signature vector increases and the signature contains more information about the characteristics of the DUT. As a result, for $N_c = 4$, we observe that both mean absolute error and error standard deviation decrease compared to $N_c = 2$.

C. Pass-fail Characterization

The predictive test can be used for pass/fail characterization of a DUT for a given performance (classification accuracy) cutoff. For an accuracy cutoff A_{cutoff}, a DUT with accuracy

Table II: Regressor prediction error statistics for VGG-16

		Nc = 2		Nc = 4	
systematic (%)	correlated (%)	MAE	STD	MAE	STD
50	0	1.71	2.51	1.63	2.39
25	25	2.38	3.39	2.12	3.02
0	50	2.54	3.95	2.1	3.4

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on September 15,2024 at 16:58:05 UTC from IEEE Xplore. Restrictions apply.

Table III: Pass/fail characterization using the predictive framework for VGG-16 (random variation is 50%)

		systematic=50%, correlated=0%				systematic=25%, correlated=25%				systematic=0%, correlated=50%			
	A_{cutoff}	70	75	80	85	70	75	80	85	70	75	80	85
Nc = 2	TP	61.8	56.2	48	38.6	78.4	73.8	65.4	48.2	90.6	86.4	78	55
	TN	35.6	39.8	48.6	57.6	17.4	22.2	29.4	41	7.2	10	14.8	29.2
	FP	0.8	1.8	1.8	1.6	2.2	2	2.6	7.4	1.2	2.2	4.8	11
	FN	1.8	2.2	1.6	2.2	2	2	2.6	3.4	1	1.4	2.4	4.8
Nc = 4	TP	62.8	56.6	46.8	38.6	78.4	74	65.4	49.4	91.2	86.8	79.2	56.6
	TN	35.2	40	49	57.4	17.6	23.2	30.2	40.4	7.6	10.2	15.2	31.2
	FP	1.2	1.6	1.4	1.8	2	1	1.8	8	0.8	2	4.4	9
	FN	0.8	1.8	2.8	2.2	2	1.8	2.6	2.2	0.4	1	1.2	3.2

Table IV: Comparison with state of the art for VGG-16

	systematic=50%, correlated=0%				systematic = 25%, correlated = 25%				systematic = 0%, correlated = 50%			
	MAE	STD	MR_{80}	MR_{85}	MAE	STD	MR_{80}	MR_{85}	MAE	STD	MR_{80}	MR_{85}
Ours	1.71	2.51	3.4	3.8	2.38	3.39	5.2	10.8	2.54	3.95	7.2	15.8
[8]	2.53	3.84	4.6	5	3.25	5.05	9.4	14.2	3.7	6.08	11.8	23
[6]	2.6	3.8	6.4	6.6	2.8	4.07	6.4	13.6	3.78	6.19	10.0	24.6

A should be characterized as "good" if $A > A_{cutoff}$, otherwise the DUT should be characterized as "bad". On the other hand, if the regressor predicts the DUT accuracy as $\hat{A}$, the predictive test characterizes a DUT as "good" if $\hat{A} > A_{cutoff}$, otherwise the DUT is characterized as "bad". As a result, a DUT can be in one of the four following categories: ① True Positive ($A > A_{cutoff}$, $\hat{A} > A_{cutoff}$) ② True Negative ($A \leq A_{cutoff}$, $\hat{A} \leq A_{cutoff}$) ③ False Positive ($A \leq A_{cutoff}$, $\hat{A} > A_{cutoff}$) ④ False Negative ($A > A_{cutoff}$, $\hat{A} \leq A_{cutoff}$) Table III shows the performance of the predictive test for a range of accuracy cutoffs for VGG-16 CNN. As A_{cutoff} increases, more devices have accuracies below cutoff, i.e., the number of bad devices increases. As a result, in Table III, as A_{cutoff} increases, true positives decrease and true negatives increase. For $A_{cutoff} = 70\%, 75\%$ and 80%, both false positives and false negatives are below 5%. For $A_{cutoff} = 85\%$ and correlated variation of 25% and 50%, false positives are above 7% for both $N_c = 2$ and 4. It is seen that false positives and negatives come only from DUTs with $|\hat{A} - A_{cutoff}| < \epsilon$, for some constant ϵ. For applications in online built-in test, this leaves scope for further improvement. On the other hand, for applications in post manufacture tests, false positives can be further reduced by applying standard test to DUTs with accuracy close to cutoff accuracy, i.e., $|\hat{A} - A_{cutoff}| < \epsilon$ (standard test refers to applying original test dataset to a DUT). However, we show in Section VI-D that our proposed compact test dataset optimization algorithm significantly outperforms prior work.

D. Comparison with Prior Work

Finally, we compare our approach to the compact test dataset generation methods proposed in [6], [8]. For comparison, we use VGG-16 CNN, fix $N_c = 2$ and simulate various conditions. For an accuracy cutoff of A_{cutoff}, we define DUT misclassification rate as $MR_{A_{cutoff}} = FP+FN$, where both false positives and false negatives are represented as percentages. Table IV shows that our proposed compact test dataset optimization algorithm outperforms the state of the art in terms of mean absolute error, error standard deviation and misclassification rate for all simulation conditions. For systematic and correlated variation of 25% each, the proposed framework reduces regression MAE by 15% (2.8% to 2.38%), error STD by 16.7% (4.07% to 3.39%) and MR_{85} by 20.5% (13.6% to 10.8%) with respect to state the state of the art. Similarly, for 50% correlated variation, the proposed framework reduces regression error by 31.3% (3.7% to 2.54%), error STD by 35% (6.08% to 3.95%) and MR_{85} by 31.3% (from 23% to 15.8%).

Table V: Results for Lenet

systematic	correlated	MAE	STD	MR_{95}	MR_{96}	MR_{97}
50	0	0.31	0.85	0.6	1.4	2.8
25	25	0.32	0.59	1.4	2	5.8
0	50	0.25	0.41	0.6	1	3.6

E. Results for MNIST

Table V shows the simulation results for Lenet CNN trained on MNIST dataset. For all simulation condition, the regressor achieves mean absolute error of less than 0.4% and error standard deviation of less than 1%. For a range of accuracy cutoffs (95-97%), the proposed test framework achieves misclassification rate of less than 6%.

F. Runtime

To evaluate the runtime of the proposed test framework, we run it using T4 GPU with two images in the compact test dataset. The entire test takes 7.8ms to complete.

VII. Conclusion

In this work, we propose a novel pixel level image optimization algorithm to create a compact test dataset for predictive testing of RRAM based CNNs. The images for backpropagation guided image optimization are initialized with a greedy search, which further improves accuracy of predictive tests. The combination of greedy down-selection and backpropagation guided optimization enables predictive testing of RRAM based CNNs with as few as 2-4 images. Given the low complexity of the proposed framework, it can be used for built-in-tests.

Acknowledgment

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

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on September 15,2024 at 16:58:05 UTC from IEEE Xplore. Restrictions apply.

REFERENCES

[1] S. Dong, P. Wang, and K. Abbas, "A survey on deep learning and its applications," *Computer Science Review*, vol. 40, p. 100379, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1574013721000198

[2] W. Sun, J. Yue, Y. He, Z. Huang, J. Wang, W. Jia, Y. Li, L. Lei, H. Jia, and Y. Liu, "A survey of computing-in-memory processor: From circuit to application," *IEEE Open Journal of the Solid-State Circuits Society*, vol. 4, pp. 25–42, 2024.

[3] D. Ielmini and H.-S. P. Wong, "In-memory computing with resistive switching devices," *Nature Electronics*, vol. 1, pp. 333 – 343, 2018.

[4] P. Chi, S. Li, C. Xu, T. Zhang, J. Zhao, Y. Liu, Y. Wang, and Y. Xie, "Prime: A novel processing-in-memory architecture for neural network computation in reram-based main memory," in *2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA)*, 2016, pp. 27–39.

[5] A. Eldebiky, G. L. Zhang, G. Böcherer, B. Li, and U. Schlichtmann, "Correctnet+: Dealing with hw non-idealities in in-memory-computing platforms by error suppression and compensation," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 43, no. 2, pp. 573–585, 2024.

[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. Meng, F. S. Hosseini, and C. Yang, "A self-test framework for detecting fault-induced accuracy drop in neural network accelerators," in *2021 26th Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2021, pp. 722–727.

[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] R. Voorakaranam, S. S. Akbay, S. Bhattacharya, S. Cherubal, and A. Chatterjee, "Signature testing of analog and rf circuits: Algorithms and methodology," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 54, no. 5, pp. 1018–1031, 2007.

[11] 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. [Online]. Available: https://www.frontiersin.org/articles/10.3389/fncom.2021.674154

[12] p. pouyan, E. Amat, and A. Rubio, "Memristive crossbar memory lifetime evaluation and reconfiguration strategies," *IEEE Transactions on Emerging Topics in Computing*, vol. 6, no. 2, pp. 207–218, 2018.

[13] H. Shin, M. Kang, and L.-S. Kim, "Fault-free: A fault-resilient deep neural network accelerator based on realistic reram devices," in *2021 58th ACM/IEEE Design Automation Conference (DAC)*, 2021, pp. 1039–1044.

[14] 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.

[15] 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.

[16] 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.

[17] 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

[18] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, A. Müller, J. Nothman, G. Louppe, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and Édouard Duchesnay, "Scikit-learn: Machine learning in python," 2018.

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

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

[21] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," 2019.

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

[23] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[24] L. Deng, "The mnist database of handwritten digit images for machine learning research," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 141–142, 2012.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on September 15,2024 at 16:58:05 UTC from IEEE Xplore. Restrictions apply.