# Adaptive Testing of Compute-in-Memory GANs Using Backpropagation-Guided Test Compaction

Anurup Saha, Ashiqur Rasul, Thomas A. Walton, Amirali Aghazadeh and Abhijit Chatterjee
School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta GA 30332
Email: asaha74@gatech.edu, arasul6@gatech.edu, twalton42@gatech.edu,
amiralia@gatech.edu, abhijit.chatterjee@ece.gatech.edu

***Abstract*—Generative adversarial networks (GANs) are promising for a range of applications, including image translation and denoising, as well as synthetic data generation. These applications can be mapped to memristive crossbar arrays (MCAs) for ultra-high energy efficiency and portability. However, conductance variation within analog crossbars degrades the quality of the GAN outputs and necessitates robust post-manufacturing testing. We propose a two-stage adaptive test framework for compute-in-memory (CiM) based GANs, comprising an exhaustive test and a compact test. The exhaustive test measures the inception score of a device under test (DUT) by applying a large number of noise vectors, called the exhaustive noise set. To reduce test time, a compact test estimates the inception score of a DUT from a carefully chosen subset of these vectors, called the compact noise set. The compact noise set is determined by a binary mask optimized with a novel backpropagation-guided algorithm to minimize the difference between the estimated and true inception scores of the DUTs. Finally, to leverage both the accuracy of the exhaustive test and the speed of the compact test, the proposed adaptive test framework first applies the compact test to every DUT. Only the DUTs that yield low confidence in classifications are then subjected to the exhaustive test. Experiments show that this adaptive approach achieves less than $1\%$ test escapes while offering up to $7.26\times$ speedup compared to exhaustive test.**

***Index Terms*—generative adversarial networks, compute-in-memory, functional test, test compaction.**

## I. Introduction

The proliferation of generative AI models has led to breakthroughs in image translation and denoising, synthetic data generation, and text generation [1]–[4]. Some commonly used models for image generation are generative adversarial networks (GAN) [5], diffusion models [6], and variational autoencoders [7]. Researchers have utilized memristive crossbar arrays (MCAs) to execute matrix multiplications directly in memory, substantially lowering the power consumption of deep learning workloads [8], [9]. Static matrix multiplications within a transformer and transpose convolutions in a GAN have been accelerated using MCAs [8], [9]. However, for MCA-based GANs, analog matrix multiplications are susceptible to conductance variations within the crossbar [10], and these variations can introduce computational errors that severely degrade the quality of the generated images. For example, as shown in Fig. 1, two MCA-based GANs generate realistic images, and the other two produce unusable, noisy results. This variability necessitates rigorous post-manufacture testing for every manufactured MCA-based GAN chip to ensure it meets quality standards after deployment.

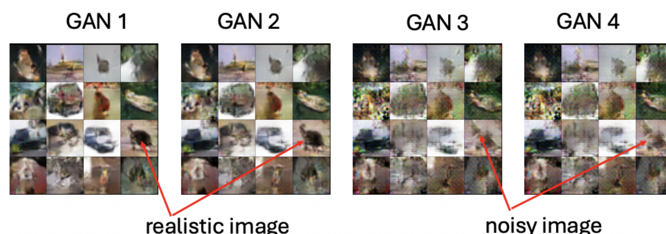


Fig. 1: Impact of conductance variation on GAN outputs

While prior work on functional testing of AI accelerators has focused on convolutional neural networks (CNNs) and spiking neural networks (SNNs) for image classification tasks and language models for text generation [11]–[17], these frameworks do not apply to image-generative models, such as GANs for two reasons: (1) The input to a GAN is a noise vector as opposed to a predefined image (2) A realistic output image from a GAN can have infinitely many real valued pixel representations as opposed to a unique class label.

In this work, we address a fundamental question: *How can we efficiently test the functional quality of MCA-based GANs*? Leveraging the Inception Score [18] as the basis for "pass"/"fail" classification, we develop an *exhaustive test* that accurately characterizes a device under test (DUT) by calculating its Inception Score from a large set of input noise vectors, called the *exhaustive noise set*. To address the high test latency of exhaustive testing, we develop a *compact test* that estimates the Inception Score using an optimized small subset of the input noise vectors, called the *compact noise set*. A backpropagation guided test compaction algorithm identifies the *compact noise set* that best predicts the Inception Score across a population of MCA-based GANs. While compact testing is fast, simulations show that compact testing alone does not achieve adequate test coverage. Finally, to achieve both fast and accurate testing, we combine these methods into an adaptive framework that first applies the rapid compact test. Only "hard-to-characterize" DUTs (i.e., their estimated inception score is close to the pass/fail threshold) are subsequently evaluated with the exhaustive test. This approach significantly reduces average test time while maintaining high test accuracy. To the best of our knowledge, this is the first research that addresses testing of MCA-based GANs.

In summary, this paper makes the following contributions: (1) We quantify the impact of crossbar conductance variation on the Inception Score of MCA-based GANs.

978-3-9826741-1-7/DATE26/© 2026 EDAA

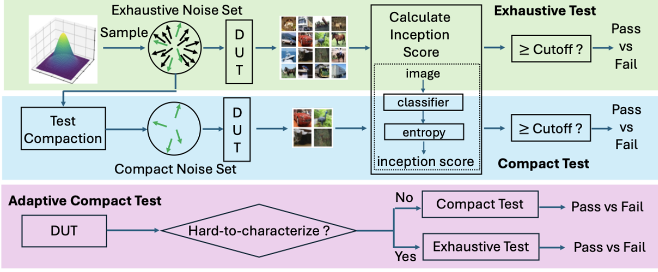


Fig. 2: Overview of the proposed test framework.

(2) We develop an exhaustive, Inception Score-based test framework for accurate functional testing.
(3) We propose a novel backpropagation-guided test compaction algorithm to drastically reduce test time.
(4) We develop a highly accurate adaptive test framework, which is up to 7.26× faster than exhaustive testing.

## II. Approach Overview

This work aims to characterize MCA-based GANs (DUTs) based on inception score (IS) [18]. If the IS of a DUT is $\Psi$ and the acceptable IS cutoff for an application is $\Psi_{th}$, a DUT is defined as "pass" if $\Psi \geq \Psi_{th}$ and as "fail" if $\Psi < \Psi_{th}$. Fig. 2 explains the overall framework, as follows:

(1) *Exhaustive Test*: In the exhaustive test framework (explained in Section IV), a large number of noise vectors sampled from the probability density function of the GAN inputs constitute the *exhaustive noise set*. In the example of Fig. 2 (top), the exhaustive noise set has 16 noise vectors. During testing, the GAN generates images corresponding to the applied noise vectors. From the generated images, the DUT's IS is calculated, and the DUT is classified as "pass" or "fail".

(2) *Compact Test*: Compact test aims to reduce the test latency by approximately estimating a DUT's IS using a small subset of the exhaustive noise set, called the compact noise set (explained in Section V). In the example shown in Fig. 2 (middle), the test compaction algorithm down-selects 4 out of the total 16 noise vectors, which are applied for estimating the DUT's IS $\hat{\Psi}$. Based on $\Psi_{th}$, the DUT is labeled as "pass" ($\hat{\Psi} \geq \Psi_{th}$) or "fail" ($\hat{\Psi} < \Psi_{th}$).

(3) *Adaptive Compact Test*: As shown in Fig. 2 (bottom), the adaptive compact test framework applies exhaustive test only to the "hard-to-characterize" DUTs. All other DUTs are classified using the compact test.

## III. Preliminaries

This work focuses on the inference phase of a popular GAN architecture: deep convolutional GAN (DCGAN) [19]. The generator in a GAN consists of transpose convolution and activation layers. Transpose convolution involves (1) upsampling the input tensors by inserting zeros and (2) convolution with the filter weights. In MCA-based architectures, the flattened weight matrix of a transpose convolution layer is stored using MCA. Upsampled inputs are fed to the crossbar, and outputs are generated along columns of the crossbar. We refer the reader to [5], [9] for more information about the mathematical details and MCA-based acceleration of GANs.

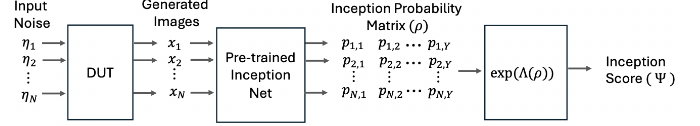


Fig. 3: Inception Score Calculation.

A major limitation of this architecture is that memristor cells within a crossbar array suffer from device-to-device and cycle-to-cycle conductance variations [10], [20]. Cycle-to-cycle conductance variation can be modeled as independent random variation. Device-to-device variation can be inter-chip or intra-chip [21], [22]. Inter-chip variation can be modeled as global or systematic variation, whereas intra-chip variation can be modeled as random or local variation. In the presence of such conductance variations, the weights represented by the crossbar differ from the GAN's trained weights. Such weight deviations introduce computation errors in matrix multiplication outputs, and these errors propagate to the image, degrading its quality.

To model the inference of the generator in the presence of such computation errors, we replace each ideal weight $w_i$ of the generator with a corresponding non-ideal weight $w_i^{'}$ as:

$$w_i^{'} = w_i(1+\epsilon_i) = w_i(1+\epsilon^{sys}+\epsilon_i^{rnd}). \quad (1)$$

Here $\epsilon_i$ refers to the non-ideality factor, and $\epsilon^{sys}$ and $\epsilon_i^{rnd}$ refer to the systematic and random non-ideality factors. By definition, all memristor cells share the same systematic non-ideality factor and is sampled from a zero-mean normal distribution with variance $\sigma_{sys}^2$. The random non-ideality factor of each memristor cell is sampled independently from another zero-mean normal distribution with variance $\sigma_{rnd}^2$. Hence, the variance of $\epsilon_i$ is $\sigma_{tot}^2 = \sigma_{sys}^2 + \sigma_{rnd}^2$. We define the percentages of systematic and random variations as $\frac{\sigma_{sys}^2}{\sigma_{tot}^2} \times 100\%$ and $\frac{\sigma_{rnd}^2}{\sigma_{tot}^2} \times 100\%$.

## IV. Exhaustive Test

*The core objective of the test framework is to assess how well the generator can produce images that resemble the distribution of natural images*. Inception Score (IS) is used as the evaluation metric because it simultaneously assesses both the visual fidelity and the diversity of the generated image distribution. Conductance variations in the crossbar arrays introduce computational errors that directly degrade the image quality and class distinguishability, measured using IS. In this section, we present the mathematical basis of IS, its calculation for a DUT using a set of input noise vectors, and the method for determining the final exhaustive noise set for testing.

### A. Mathematical Formulation of Inception Score

To calculate the IS of a GAN, its output images are passed through Inception v3 [23], an Inception network pretrained on the ImageNet dataset [24]. Assume that a GAN generates an image $x$ corresponding to an input noise vector $\eta$ (as shown in Fig. 3). For every image $x$, Inception net assigns class conditional probabilities $\mathbb{P}(y|x)$ for a total of $Y$ output classes. IS is formally defined as the exponential of the Kullback-Leibler (KL) divergence [25] between the class conditional distribution for a generated image $\mathbb{P}(y|x)$, and the marginal

class distribution $\mathbb{P}(y)$ over the set of all generated images [18]:

$$\Psi = \exp\left(\mathbb{E}_x[D_{KL}(\mathbb{P}(y|x))\|\mathbb{P}(y)]\right) \tag{2}$$

If the image $x$ is of good quality, Inception net assigns a high probability to only one class, resulting in low entropy of $\mathbb{P}(y|x)$. On the other hand, if the generated images have high diversity, $\mathbb{P}(y)$ has high entropy. We show that the logarithm of IS (referred to as the log-IS) is the difference of the entropy of $\mathbb{P}(y)$ (called *sample entropy*) and the entropy of $\mathbb{P}(y|x)$ (called *class conditional entropy*). The log-IS can be expressed as:

$$\begin{aligned}
\alpha &= \log\Psi = \mathbb{E}_x\big[D_{KL}\big(\mathbb{P}(y|x)||\mathbb{P}(y)\big)\big]\\
&= \mathbb{E}_x\sum_{y=1}^{Y}\mathbb{P}(y|x)\log\frac{\mathbb{P}(y|x)}{\mathbb{P}(y)}\\
&= -\mathbb{E}_x\sum_{y=1}^{Y}\mathbb{P}(y|x)\log\mathbb{P}(y) + \mathbb{E}_x\sum_{y=1}^{Y}\mathbb{P}(y|x)\log\mathbb{P}(y|x)\\
&= -\sum_{y=1}^{Y}\sum_{x}\mathbb{P}(y|x)\mathbb{P}(x)\log\mathbb{P}(y) + \mathbb{E}_x\sum_{y=1}^{Y}\mathbb{P}(y|x)\log\mathbb{P}(y|x)\\
&= -\sum_{y=1}^{Y}\mathbb{P}(y)\log\mathbb{P}(y) + \mathbb{E}_x\sum_{y=1}^{Y}\mathbb{P}(y|x)\log\mathbb{P}(y|x) \qquad (3)\\
&= H_s - H_c \qquad (4)
\end{aligned}$$

In (3), the first term is referred to as the sample entropy $H_s$ and the second term is referred to as the class conditional entropy $H_c$ as defined in (7) and (9).

### B. Log-Inception Score Calculation of a DUT

We now explain how the sample and class conditional entropy of a DUT can be calculated by applying a set of $N$ noise vectors to the DUT. Assume that we apply $N$ noise vectors $\{\eta_1, \eta_2, \cdots, \eta_N\}$ to a DUT (as shown in Fig. 3). Corresponding to the $n$-th noise vector, the generated image is $x_n$. When $x_n$ is applied to the pretrained Inception net, the output probability corresponding to the $y$-th class is $\mathbb{P}(y = y|x = x_n) = p_{n,y}$. The *Inception probability matrix* with respect to the set of $N$ the applied noise vectors $\{\eta_n\}_{n=1}^{N}$ is defined as:

$$\rho = \begin{bmatrix} p_{1,1} & p_{1,2} & \cdots & p_{1,Y}\\ \vdots & \vdots & \vdots & \vdots\\ p_{N,1} & p_{N,2} & \cdots & p_{N,Y}\end{bmatrix} \tag{5}$$

Since each image $x_n$ in the set $\{x_n\}_{n=1}^{N}$ has equal probability of being generated, $\mathbb{P}(x_n) = \frac{1}{N}$. From $\rho$ and $\mathbb{P}(x_n)$, we can calculate the sample and class conditional entropies as:

$$\mathbb{P}(y) = \sum_{n=1}^{N}\mathbb{P}(y|x_n)\mathbb{P}(x_n) = \frac{1}{N}\sum_{n=1}^{N}p_{n,y} \tag{6}$$

$$H_s = -\sum_{y=1}^{Y}\mathbb{P}(y)\log\mathbb{P}(y) \tag{7}$$

$$= -\sum_{y=1}^{Y}\big(\frac{1}{N}\sum_{n=1}^{N}p_{n,y}\big)\big(\log\frac{1}{N}\sum_{n=1}^{N}p_{n,y}\big) \tag{8}$$

$$H_c = -\mathbb{E}_x\sum_{y=1}^{Y}\mathbb{P}(y|x)\log\mathbb{P}(y|x) \tag{9}$$

$$= -\sum_{n=1}^{N}\mathbb{P}(x_n)\sum_{y=1}^{Y}p_{n,y}\log p_{n,y} = -\frac{1}{N}\sum_{n=1}^{N}\sum_{y=1}^{Y}p_{n,y}\log p_{n,y} \tag{10}$$

In summary, from an $N \times Y$ dimensional *Inception probability matrix*, the log-IS $\alpha$ is calculated as:

$$\begin{aligned}\alpha = \Lambda(\rho) = -\sum_{y=1}^{Y}\big(\frac{1}{N}\sum_{n=1}^{N}p_{n,y}\big)\log\big(\frac{1}{N}\sum_{n=1}^{N}p_{n,y}\big)\\ + \frac{1}{N}\sum_{n=1}^{N}\sum_{y=1}^{Y}p_{n,y}\log p_{n,y}\end{aligned} \tag{11}$$

Here, $\Lambda$ is defined as a function that converts the Inception probability matrix of a DUT to its log-IS.

### C. Overall Exhaustive Test

The IS of a GAN depends on the set of input noise vectors. If $J$ independently sampled sets of $N$ noise vectors $\{T^j\}_{j=1}^{J}$ (each $T^j$ has $N$ noise vectors) are applied to a GAN and the measured inception scores corresponding to the noise sets are $\{\Psi^j\}_{j=1}^{J}$, then $\Psi^i \neq \Psi^j$ for $i \neq j$. To fix the size of the exhaustive noise set, we want to set $N$ sufficiently large so that the standard deviation $\sigma$ and mean $\mu$ of $\{\Psi^j\}_{j=1}^{J}$ satisfy $\frac{\sigma}{\mu} < 1\%$. Once $N$ is fixed, $N$ noise vectors $\{\eta\}_{n=1}^{N}$ are sampled from $\mathbb{P}(\eta)$, which constitute the exhaustive noise set $\mathscr{E}$.

The overall exhaustive test can be summarized as:
(1) Create an exhaustive noise set $\mathscr{E} = \{\eta_1, \eta_2, \cdots, \eta_N\}$.
(2) Apply each noise vector from $\mathscr{E}$ to a DUT and from DUT outputs create the Inception probability matrix $\rho \in \mathbb{R}^{N\times Y}$.
(3) Calculate log-inception score $\alpha$ using (11). Calculate the DUT's inception score $\Psi = \exp(\alpha)$, and label it as "pass" if $\Psi \geq \Psi_{th}$ for a predefined threshold $\Psi_{th}$.

## V. Compact Test

The exhaustive test described in Section IV is guaranteed to correctly label a DUT as "pass"/"fail". However, it incurs a significant test time cost due to the large number of noise vectors ($N$) required. To accelerate the testing process, we introduce a compact test that estimates the IS using a much smaller, carefully selected subset of $K$ noise vectors ($K < N$), referred to as the compact noise set $\mathscr{C}$. The key challenge is to select a compact noise set that is representative of the exhaustive noise set.

### A. Optimization Objective

For test compaction, we first apply the exhaustive test to initial $D$ DUTs. The Inception probability matrix of the $d$-th DUT is computed as $\rho^d$ and the log-IS as $\alpha^d$. Our goal is to choose a compact noise set $\mathscr{C}$ such that if the log-IS of the $d$-th DUT measured using $\mathscr{C}$ is $\hat{\alpha}^d$, then $\alpha^d \approx \hat{\alpha}^d$. To down-select noise vectors, we introduce a binary mask $M = [m_1 \quad m_2 \quad \cdots \quad m_N]$, where $m_n = 1$ indicates that $\eta_n \in \mathscr{C}$. To pick a compact noise set with $K$ noise vectors, the test compaction problem can be mathematically formulated as:

$$\min_{M} \frac{1}{D} \sum_{d=1}^{D} |\alpha^d - \hat{\alpha}^d| \text{ subject to} \tag{12}$$

$$m_n \in \{0, 1\} \quad \forall 1 \leq n \leq N \tag{13}$$

$$\sum_{n=1}^{N} m_n = K. \tag{14}$$

This is a combinatorial optimization problem with $\binom{N}{K}$ possible solutions. Since $K$ can be of the order of $10^3$, stochastic optimization methods, such as a genetic algorithm, are computationally intractable. Next, we introduce a backpropagation-guided test compaction algorithm, which leverages the modern deep learning framework PyTorch [26] and can be accelerated using GPUs to optimize the binary mask.

### *B. Soft-mask Based Optimization*

The core idea of the constrained optimization framework is to introduce a "soft-mask", namely a neural network with noise input signals as network inputs and class probabilities as outputs. This is updated using gradient-based optimization and then binarized to obtain $M$. We introduce a set of trainable real-valued weights, $W = [w_1 \quad w_2 \quad \dots \quad w_N]$, one corresponding to each noise vector in $\mathscr{E}$. A softmax function converts these weights into the soft-mask, $\tilde{M} = [\tilde{m}_1 \quad \tilde{m}_2 \quad \cdots \quad \tilde{m}_N]$, where each element is calculated as:

$$\tilde{m}_n = K \times \frac{\exp(w_n)}{\sum_{n'=1}^{N} \exp(w_{n'})}. \tag{15}$$

Equation (15) ensures $\sum_{n=1}^{N} \tilde{m}_n = K$ and the mask is differentiable with respect to the $W$. Accounting for the soft-mask, the marginal class distribution $\mathbb{P}(y)$, sample entropy $H_s$, class conditional entropy $H_c$ and log-IS $\alpha$ are calculated as:

$$\mathbb{P}(y) = \frac{\sum_{n=1}^{N} \tilde{m}_n p_{n,y}}{\sum_{n=1}^{N} \tilde{m}_n} \tag{16}$$

$$H_s = -\sum_{y=1}^{Y} \mathbb{P}(y) \log \mathbb{P}(y) \tag{17}$$

$$H_c = -\sum_{n=1}^{N} \frac{\tilde{m}_n}{\sum_{n=1}^{N} \tilde{m}_n} \left( \sum_{y=1}^{Y} p_{n,y} \log p_{n,y} \right) \tag{18}$$

$$\hat{\alpha} = H_s - H_c \tag{19}$$

### *C. Test Compaction Algorithm*

The test compaction algorithm (outlined in Algorithm 1) starts with total $D$ DUTs for which the Inception probability matrices corresponding to the exhaustive noise set and inception scores are known. In each epoch, the training is split into $\frac{D}{B}$ batches, where in each batch $B$ randomly sampled DUTs are used. For each DUT, based on the current weight, the log-IS is estimated as $\hat{\alpha}^b$. The loss is calculated as the mean absolute error between the true and estimated IS. The gradient of the loss with respect to the trainable weights is calculated, and the weights are updated using the Adam optimizer. At the end of the $e$-th epoch, the weights are updated such that the soft-mask contains $K^{'}$ ones and $N - K^{'}$ zeros. $K^{'}$ is reduced from $N$ to $K$ using a multiplicative scheduler.

**Algorithm 1** Overall Test Compaction Framework

1: **Input:** $D$ DUTs with known inception probability matrices and true log-IS $\{\rho^d, \alpha^d\}_{d=1}^{D}$ from the exhaustive test. Size of the exhaustive noise set = $N$ and compact noise set = $K$. Number of epochs $E$, batch size = $B$.
2: **Output:** Binary mask $M = [m_1 \quad m_2 \quad \cdots \quad m_N]$.
3: Set $K^{'} = N$ and initialize $W = [w_1 \quad w_2 \quad \dots \quad w_N]$ randomly.
4: **for** $e = 1$ to $E$ **do**
5: Calculate soft-mask $\tilde{M} = [\tilde{m}_1 \quad \tilde{m}_2 \quad \cdots \quad \tilde{m}_N]$ where $\tilde{m}_n = K^{'} \times \frac{\exp(w_n)}{\sum_{t=1}^{N} \exp(w_t)}$.
6: **for** batch = 1 to $\frac{D}{B}$ **do**
7: Sample $B$ out of total $D$ DUTs. Assume the $b$-th DUT has Inception probability matrix $\rho^b$ and log-IS $\alpha^b$ $(1 \leq b \leq B)$.
8: Based on the soft-mask, estimate the log-IS of the $b$-th DUT as $\hat{\alpha}^b$.
9: Calculate loss $\mathcal{L} = \frac{1}{B} \sum_{b=1}^{B} |\alpha^b - \hat{\alpha}^b|$.
10: Calculate gradients $\nabla_W \mathcal{L}$.
11: Update weights: $W \leftarrow f_{adam}(W, \nabla_W \mathcal{L})$.
12: **end for**
13: Set $K^{'} = K^{\frac{e}{E}} N^{1-\frac{e}{E}}$
14: Set top-$K^{'}$ weights of $W$ to 0 and other weights to $-\infty$
15: **end for**
16: Calculate $\tilde{M}$ and set $M = \tilde{M}$
17: **Return** $M$

## VI. Adaptive Compact Test

Despite being $\frac{N}{K}\times$ faster than exhaustive test, compact test can misclassify DUTs. We observe that if the IS of a DUT is $\Psi$ and the cutoff IS is $\Psi_{th}$, then a smaller value of $|\Psi - \Psi_{th}|$ results in a higher likelihood of the DUT being misclassified. The proposed adaptive compact test framework has two stages:
(1) A DUT's IS is first estimated from the compact test. For a predefined uncertainty range $\Delta$, the DUT is determined as hard-to-characterize if the estimated IS $\hat{\Psi}$ satisfies $|\hat{\Psi} - \Psi_{th}| \leq \Delta$. Otherwise the DUT is classified as "pass" ($\hat{\Psi} \geq \Psi_{th}$) or "fail" ($\hat{\Psi} < \Psi_{th}$).
(2) If a DUT is hard-to-characterize, all the remaining noise vectors from the set $\mathscr{E} - \mathscr{C}$ are applied to the DUT and its true IS $\Psi$ is calculated. Based on $\Psi$ and $\Psi_{th}$, the DUT is labeled as "pass" or "fail".

To calculate $\Delta$, we use the $D$ DUTs used for test compaction. If the true and the estimated IS of the $d$-th DUT is $\Psi^d$ and $\hat{\Psi}^d$, then we define the *uncertainty range* $\Delta$ as:

$$\Delta = \nu \times \max_{d=\{1,2,\cdots,D\}} |\Psi^d - \hat{\Psi}^d| \tag{20}$$

We call $\nu (\nu > 0)$ the *uncertainty range multiplier*. For larger values of $\nu$, more DUTs undergo exhaustive test.

## VII. Results

### *A. Experimental Setup*

We evaluate MCA based deep convolutional GAN (DC-GAN) [19] trained using CIFAR-10 and CIFAR-100 datasets [27]. The baseline inception score (IS) is 6.47 for CIFAR-10 and 6.70 for CIFAR-100. Following prior work, we restrict maximum systematic variation to 50% of the total variation [21]. We simulate for 3 variability conditions: (1) typical variation (25% systematic and 75% random) (2) extreme systematic variations (50% systematic and 50% random) and (3) extreme random variation (0% systematic and 100% random).

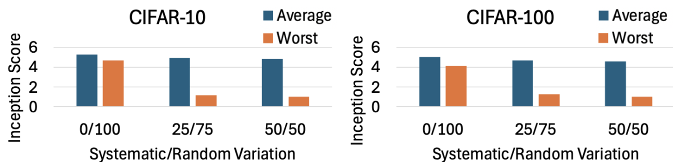


Fig. 4: Inception score vs variability trends.

We assume $\sigma_{tot} = 0.3$ (defined in Section III) and input noise vectors ($\eta \in \mathbb{R}^{100}$) follow normal distribution, i.e., $\eta \sim \mathcal{N}(0, I)$. For test compaction and uncertainty range $\Delta$ determination, we use $D = 1000$ DUTs. We evaluate the compact test and the adaptive compact test on another 1000 DUTs. For test compaction, we optimize for total $E = 100$ epochs using the Adam optimizer [28] in Pytorch [26]. For calculating the inception probability matrix, gan-metrics-pytorch repository [29] is used. We observe that $\Psi_{th} > 6.0$ results in low manufacturing yield and $\Psi_{th} < 4.5$ leads to deployments of GANs which generate blurry images. Hence, we evaluate the proposed framework for cut-off inception scores in the range $4.5 \leq \Psi_{th} \leq 6.0$. The framework is evaluated using 3 metrics:

(1) Test Escape (TE): If the inception score (IS) of a DUT $\Psi$ is less than the cutoff IS of $\Psi_{th}$, and the DUT is classified as "pass", it is referred to as test escape.

(2) Yield Loss (YL): If a DUT is classified as "fail", whereas $\Psi \geq \Psi_{th}$, it is referred to as yield loss.

(3) Test Speedup: It measures how fast a test framework is compared to the exhaustive test. During testing, if the $d$-th DUT requires $N_d$ input noise vectors, we calculate Test Speedup $= \frac{N}{\frac{1}{1000}\sum_{d=1}^{1000} N_d}$. For compact test, $N_d = K$ and the speedup is $\frac{N}{K}$. The overall objective is to achieve low test escape, low yield loss, and high test speedup. Throughout this section, we report test escape and yield loss as percentages.

### B. Impact of Variability

For each variability condition, we simulate 1000 DUTs and calculate their average and minimum (worst-case) IS. Fig. 4 shows that for CIFAR-10, the average and worst IS for 50% systematic variation are 4.85 and 1.0. For 0% systematic variation the average and worst case IS are 5.27 and 4.66. For CIFAR-100, as we go from 50% to 0% systematic variation, the average IS increases from 4.61 to 5.04 and the worst-case IS increases from 1.00 to 4.13. *The key takeaway is that systematic variation is more detrimental to image generation quality of GANs than random variation.*

### C. Determine the Size of Exhaustive Noise Set

Section IV-C explains how to determine the size of the exhaustive noise set $N$. We generate $J = 100$ independent sets of $N$ noises and measure the IS of the trained GAN model for each set of input noises. For each $N$, we calculate the mean ($\mu$) and standard deviation ($\sigma$) of the measured inception scores. Table I shows that for both CIFAR-10 and CIFAR-100, we achieve $\sigma/\mu < 1\%$ for $N = 10000$. So we use $N = 10000$ noise vectors for exhaustive testing.

### D. Performance of Compact Test

Fig. 5 and Fig. 6 respectively, show the test escape and yield loss for the compact test framework. For typical variation (25% systematic, 75% random) and $\Psi_{th}$ in the range $4.5 - 6.0$, test escape lies within $0.4 - 5.2\%$ for $K = 500$ ($K$ is the number of noise vectors for compact test), $0.3 - 4\%$ for $K = 1000$ and $0.2 - 3.2\%$ for $K = 2000$. Similarly, yield loss lies within $0.4 - 7.2\%$ for $K = 500$, $0.4 - 4.6\%$ for $K = 1000$ and $0.1 - 3.3\%$ for $K = 2000$. As $K$ increases, the inception score estimates become more accurate leading to lower test escapes and yield loss. For $K = 1000$ and across different variability conditions, the worst case test escape is 10.2% for $\Psi_{th} = 5.3$, 100% random variation and the worst case yield loss is 7.4% for $\Psi_{th} = 5.2$, 100% random variation. It is observed that for 100% random variation the test escapes and yield loss are higher compared to other variability conditions. Results for the CIFAR-100 dataset show similar trends. *The key takeaways are: (a) Higher $K$ leads to lower test escapes and yield loss, but lower test speedup (b) For a given $K$, 100% random variation has the highest test escape and yield loss.*

TABLE I: Ratio of standard deviation to mean ($\sigma/\mu$) vs $N$

| $N$ | | 500 | 1000 | 2000 | 5000 | 10000 |
|---|---|---|---|---|---|---|
| $\sigma/\mu(\%)$ | CIFAR-10 | 3.41 | 2.49 | 1.93 | 1.3 | 0.93 |
| $\sigma/\mu(\%)$ | CIFAR-100 | 4.32 | 2.99 | 2.25 | 1.33 | 0.84 |

### E. Performance of Adaptive Compact Test

Based on the takeaways from the compact test, for an optimal trade-off between speedup and DUT mis-classification, we fix $K = 2000$ for 100% random variation and $K = 1000$ for 50% and 25% systematic variation. Fig. 7 and Fig. 8 show the test escape and yield loss for the uncertainty range multiplier $\nu = 0.6, 0.8, 1.0$ (refer to (20)). For CIFAR-10, test escape and yield loss are less than 2.1% and 1.2% for $\nu = 0.6$, 1.2% and 0.6% for $\nu = 0.8$ and 0.5% and 0.3% for $\nu = 1.0$ across all $\Psi_{th}$ and variability conditions. Similarly for CIFAR-100 and $\nu = 1.0$, test escape and yield loss are within 0.4% and 0.3%. For practical use case, $\nu = 0.8$ or 1.0 can be used. Fig. 9 shows that for CIFAR-10, $\nu = 0.8$ and different $\Psi_{th}$, speedup of adaptive compact test is $3.21 - 7.26\times$ for 75% random variation, $1.64 - 5\times$ for 100% random variation and $3.45 - 7.82\times$ for 50% random variation. For CIFAR-100 as well, we observe similar trends. For CIFAR-10, 100% random variation and $\Psi_{th} = 5.2, 5.3$ (Fig. 9b), we observe large number of hard-to-characterize DUTs resulting in lower test speedup.

### F. Test Time

For 25% systematic variation, $\Psi_{th} = 5.5$ and $\nu = 0.8$, adaptive compact test requires 0.93 seconds to test each DUT, when the Inception Net is deployed on an NVIDIA A100 GPU.

## VIII. Conclusion

We develop an adaptive compact test methodology for MCA based GANs. The test compaction is achieved using a novel backpropagation-guided binary mask optimization. The proposed adaptive compact test achieves less than 1% test escapes while demonstrating up to 7.26× test speedup compared to the exhaustive test. In future, the proposed test compaction algorithm can be used for other deep learning accelerators.

## Acknowledgement

This research was supported by the U.S. National Science Foundation under Grant: 2414361.

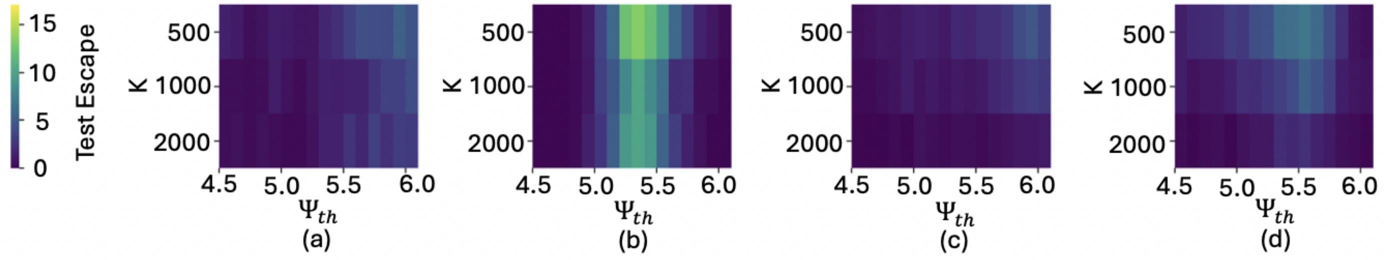


Fig. 5: Compact test: test escape vs size of compact noise set ($K$) and inception score cutoff ($\Psi_{th}$) for (a) CIFAR-10, 75% random (b) CIFAR-10, 100% random (c) CIFAR-10, 50% random (d) CIFAR-100, 75% random

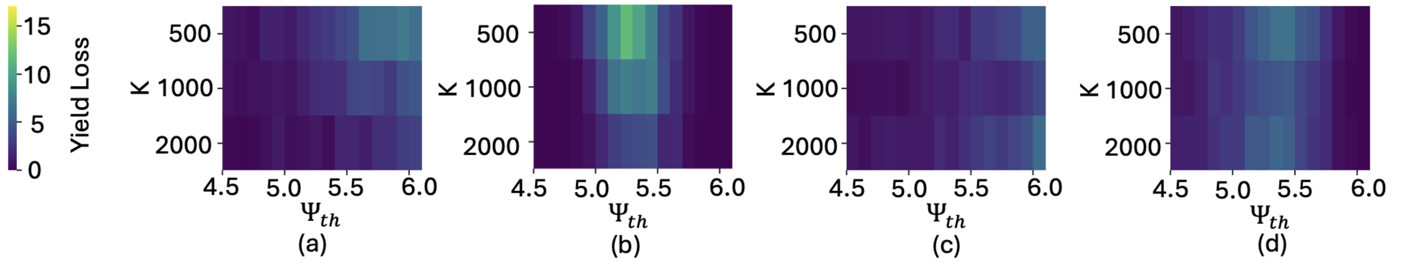


Fig. 6: Compact test: yield loss vs size of compact noise set ($K$) and inception score cutoff ($\Psi_{th}$) for (a) CIFAR-10, 75% random (b) CIFAR-10, 100% random (c) CIFAR-10, 50% random (d) CIFAR-100, 75% random

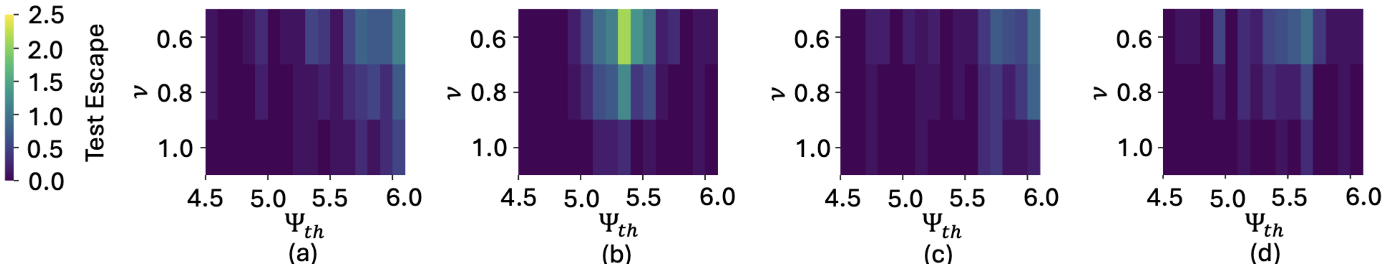


Fig. 7: Adaptive compact test: test escape vs uncertainty range multiplier ($\nu$) and inception score cutoff ($\Psi_{th}$) for (a) CIFAR-10, 75% random (b) CIFAR-10, 100% random (c) CIFAR-10, 50% random (d) CIFAR-100, 75% random

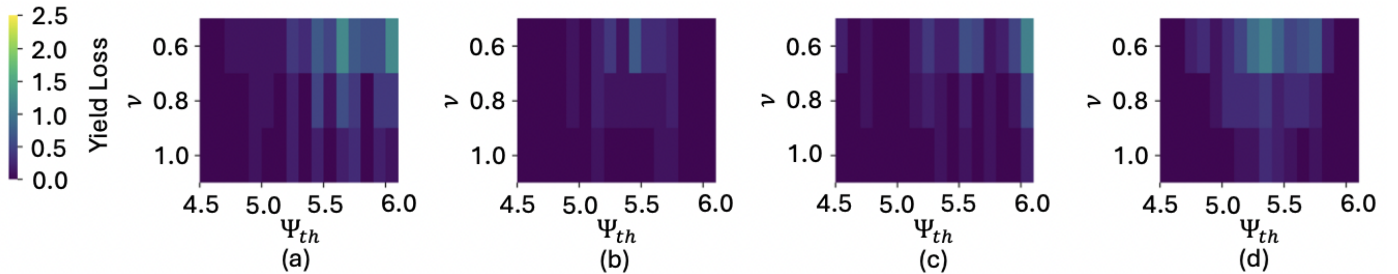


Fig. 8: Adaptive compact test: yield loss vs uncertainty range multiplier ($\nu$) and inception score cutoff ($\Psi_{th}$) for (a) CIFAR-10, 75% random (b) CIFAR-10, 100% random (c) CIFAR-10, 50% random (d) CIFAR-100, 75% random

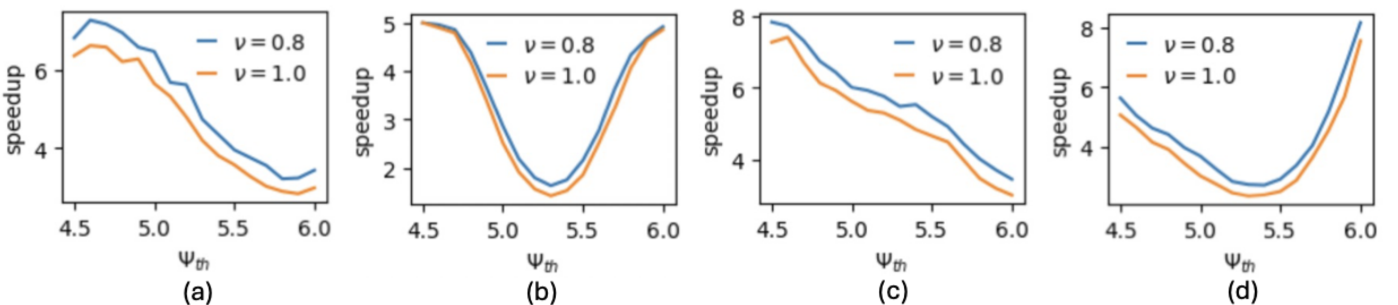


Fig. 9: Adaptive compact test: speedup vs inception score cutoff ($\Psi_{th}$) for confidence range multiplier $\nu = 0.8, 1.0$ and for (a) CIFAR-10, 75% random (b) CIFAR-10, 100% random (c) CIFAR-10, 50% random (d) CIFAR-100, 75% random

# REFERENCES


[1] S. Yan, C. Wang, W. Chen, and J. Lyu, "Swin transformer-based gan for multi-modal medical image translation," *Frontiers in Oncology*, vol. 12, p. 942511, 2022.

[2] A. Alsaiari, R. Rustagi, M. M. Thomas, A. G. Forbes *et al.*, "Image denoising using a generative adversarial network," 2019.

[3] M. Frid-Adar, E. Klang, M. Amitai, J. Goldberger, and H. Greenspan, "Synthetic data augmentation using gan for improved liver lesion classification," in *2018 IEEE 15th International Symposium on Biomedical Imaging (ISBI 2018)*, 2018, pp. 289–293.

[4] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever *et al.*, "Language models are unsupervised multitask learners," *OpenAI blog*, vol. 1, no. 8, p. 9, 2019.

[5] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," 2014. [Online]. Available: https://arxiv.org/abs/1406.2661

[6] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," 2020. [Online]. Available: https://arxiv.org/abs/2006.11239

[7] D. P. Kingma, M. Welling *et al.*, "Auto-encoding variational bayes," 2013.

[8] S. Sridharan, J. R. Stevens, K. Roy, and A. Raghunathan, "X-former: In-memory acceleration of transformers," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 31, no. 8, pp. 1223–1233, 2023.

[9] F. Chen, L. Song, and Y. Chen, "Regan: A pipelined reram-based accelerator for generative adversarial networks," in *2018 23rd Asia and South Pacific Design Automation Conference (ASP-DAC)*. IEEE, 2018, pp. 178–183.

[10] 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.

[11] 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.

[12] 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.

[13] S. Kundu, S. Banerjee, A. Raha, S. Natarajan, and K. Basu, "Toward functional safety of systolic array-based deep learning hardware accelerators," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 29, no. 3, pp. 485–498, 2021.

[14] A. Chaudhuri, C. Chen, and K. Chakrabarty, "Recent advances in testing techniques for ai hardware accelerators," *Foundations and Trends® in Integrated Circuits and Systems*, vol. 2, no. 4, pp. 244–380, 2023.

[15] A. Saha, C. Amarnath, K. Ma, and A. Chatterjee, "Confidence driven compact testing of compute-in-memory based language models," in *2025 IEEE European Test Symposium (ETS)*, 2025, pp. 1–6.

[16] A. Ruospo, G. Gavarini, A. Porsia, M. S. Reorda, E. Sanchez, R. Mariani, J. Aribido, and J. Athavale, "Image test libraries for the on-line self-test of functional units in gpus running cnns," in *2023 IEEE European Test Symposium (ETS)*, 2023, pp. 1–6.

[17] D. A. Moussa, M. Hefenbrock, and M. Tahoori, "Compressed test pattern generation for deep neural networks," *IEEE Transactions on Computers*, vol. 74, no. 1, pp. 307–315, 2025.

[18] S. Barratt and R. Sharma, "A note on the inception score," *arXiv preprint arXiv:1801.01973*, 2018.

[19] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," 2016. [Online]. Available: https://arxiv.org/abs/1511.06434

[20] 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)*. IEEE, 2019, pp. 1769–1774.

[21] 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.

[22] 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.

[23] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," 2014. [Online]. Available: https://arxiv.org/abs/1409.4842

[24] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2009, pp. 248–255.

[25] S. Kullback and R. A. Leibler, "On information and sufficiency," *Annals of Mathematical Statistics*, vol. 22, no. 1, pp. 79–86, 1951.

[26] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, 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," in *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché Buc, E. Fox, and R. Garnett, Eds., vol. 32. Curran Associates, Inc., 2019, pp. 8024–8035.

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

[28] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2017. [Online]. Available: https://arxiv.org/abs/1412.6980

[29] gan-metrics-python. [Online]. Available: https://github.com/abdulfatir/gan-metrics-pytorch