Off-Chip Super-Resolution AI Model to Support Embedded Memory Diagnosis

# Off-Chip Super-Resolution AI Model to Support Embedded Memory Diagnosis

Simone Anedda[1], Paolo Bernardi[1], Matteo Coppetta[2], Giorgio Insinga[1], Tommaso Montedoro[1], Annachiara Ruospo[1], Rudolf Ullmann[2], Felix Tengler[2]
[1]Politecnico di Torino, Italy
[2]Infineon Technologies, Germany and Italy

***Abstract*—Efficiently storing memory test outcomes by preserving exact failure information in Automotive Systems-on-Chip is currently one of the major test challenges. Due to the limited availability of the on-chip buffer memory and the non-negligible time needed to transmit test results, their exact failing coordinates may be wasted due to the buffer saturation. To overcome this limitation, an efficient approach is to compress the detected faults using on-chip test resources, enabling the tester to retrieve the fault information at the end of the memory test. Although this method achieves significant memory savings, it does so at the cost of losing the exact failing bitmap coordinates, thereby introducing a lossy compression effect. To solve this issue, this paper proposes a framework relying on super-resolution (SR) artificial intelligence models to reconstruct off-chip, with high accuracy, the failing bitmaps. On-chip 256x256 coordinate-based failing bit test results are saved in a compressed format using 16x16 bitmaps. This compressed diagnostic information is then fed off-chip to the SR framework to return very accurate full-chip predictions on 256x256 bitmaps. As a case of study, the developed SR model has been trained on a dataset composed of real industrial failing RRAM results, achieving 98% accuracy. Overall, the off-chip proposed methodology achieves high global SR scores: competitive metrics adopting Structural Similarity Index Measure (SSIM) have been used, and all return results above 98% accuracy of reconstruction.**

***Index Terms*—Fault Injection, Reliability, Convolutional Neural Network, Bit-Flip, Online Test.**

## I. Introduction

Testing automotive System-on-Chips (SoCs) is essential to ensure that each device meets its specifications. Embedded on-chip memories occupy a substantial portion of the die in automotive systems and thus have a major impact on yield. As a result, manufacturers perform extensive testing to collect detailed diagnostic information, whose results are stored in the available on-chip memory of the design under test (DUT). These testing outcomes support technology experts in identifying weaknesses, improving memory resiliency, and implementing countermeasures.

Recent research has investigated how to store memory test results more efficiently. Indeed, traditional diagnostic schemes such as SLAC [1], [2] ensure full accuracy by preserving the exact coordinates of each individual failing memory cell. Thus, at the end of the test, the failing bitmap of the entire memory can be reconstructed without any loss in precision (at a bit level). However, in the case of numerous faults, not all diagnostic information can be saved using lossless encoding algorithms [3]: it may saturate the available on-chip storage under fault-dense conditions.

To address this limitation, the pixel-based encoding introduced in [3] provides a lossy alternative that significantly reduces storage requirements while retaining informative spatial patterns. Instead of storing exact failing coordinates, a cumulative counter of failing cells for a specific memory region (referred to as *Pixel*) is stored. A pixel represents the fundamental unit of the proposed encoding scheme in [3]. It aggregates the faults within a fixed-size region of the memory array, storing only the region's coordinates and the corresponding fault count. The primary limitation of pixel-based encodings is the loss of bit-level granularity, meaning the precise locations of failing memory cells cannot be determined.

This work is intended to address this problem by taking inspiration from similar issues faced when reconstructing missing information in satellite images, i.e., leveraging Artificial Intelligence (AI) models. Inspired by the effectiveness of Super Resolution (SR) AI models in improving image quality (e.g., [4], [5], [6]), this work proposes a framework relying on SR techniques to off-chip reconstruct faulty memory bitmaps from pixel-based lossy encodings, *preventing saturated buffer memory from causing test results to be lost.* Image SR is a process that aims to reconstruct a high-resolution image with a low-resolution input [7]. In this work, during on-chip test execution, diagnostic data are stored using the pixel encoding, which aggregates failures within fixed-size regions (losing the exact failing coordinates but saving memory space and avoiding memory saturation); these compressed representations are referred to as Low-Resolution (LR) bitmaps. Then, the proposed reconstruction stage recovers off-chip (at the end of the test) the entire faulty memory bitmap with very high accuracy for a visual inspection of failing patterns. In this way, we do not risk saturating the on-chip memory buffer and, at the same time, guarantee that the faulty patterns are preserved. Hereinafter, the corresponding predicted non-compressed bitmap is referred to as a High-Resolution (HR) bitmap. To the best of the authors' knowledge, this represents the first time an SR AI model is used for reconstructing diagnostic data information in the field of memory tests.

The entire framework has been validated using a set of real failing bitmaps coming from the Infineon's AURIX™ TC4x

979-8-3195-1763-0/26/$31.00 ©2026 IEEE

family of microcontrollers [8].

Results show that the approach is highly effective for macroscopic inspection of faulty RRAM memories, providing, with very high accuracy (around 98%), diagnostic information about failing memory shapes.

The remainder of this paper is organized as follows. Section II reviews related work and outlines the motivation for this study. Section III describes the proposed reconstruction methodology. Section IV reports the experimental results and analysis. Finally, Section V summarizes the contributions of this work.

## II. Background and Related Work

This section intends to provide background information on compressed diagnostic memory encodings (Section II-A) and SR AI models (Section II-B).

### A. Saving compressed diagnostic information

Diagnostic analysis of embedded memories often requires collecting large amounts of failing-bit information. Traditional bitmaps store the exact coordinates of each faulty cell, but this becomes impractical when the number of faults grows or when on-chip diagnostic memory is limited. To address this issue, [3] proposes a pixel-based encoding that provides a compact and topologically meaningful representation of memory failures. The method groups adjacent wordlines and bitlines into fixed-size regions called *pixels*. For each pixel, only three values are stored: its $x_{\text{pixel}}$ coordinate, its $y_{\text{pixel}}$ coordinate, and a counter ($count$) of the faults occurring inside that region. This reduces the stored data to a small four-byte structure per pixel and scales with the fault density rather than the memory size. Pixels are created only when a fault is detected inside their region, enabling efficient representation even in fault-dense scenarios. Because each pixel accumulates failures from its area, the resulting map preserves the topological distribution of faults and helps identify regions with abnormal behavior during early characterization.

The pixel size is configurable, allowing designers to balance resolution and memory usage. Larger pixels increase compression, while smaller ones provide more spatial detail. Since the representation focuses on fault density, it is compatible with both hardware- and software-based memory test strategies and can be enabled when on-chip diagnostic memory becomes constrained. The stored pixels can be visualized offline as a color-coded map, in which the color reflects the normalized number of faults in each pixel. This view allows fault clusters to be easily recognized and supports analysis of memory behavior under varying test conditions.

To compute the pixel, the memory array is divided into square regions of size $\texttt{bitsPerPixel} \times \texttt{bitsPerPixel}$. Each detected fault is identified by its physical coordinates:

- $x_{\text{Fault}}$: bitline index;
- $y_{\text{Fault}}$: wordline index.

The corresponding pixel coordinates are computed using integer division:

$$x_{\text{Pixel}} = \left\lfloor \frac{x_{\text{Fault}}}{\texttt{bitsPerPixel}} \right\rfloor, \tag{1}$$

$$y_{\text{Pixel}} = \left\lfloor \frac{y_{\text{Fault}}}{\texttt{bitsPerPixel}} \right\rfloor. \tag{2}$$

All faults falling within the same pixel region share the same $(x_{\text{Pixel}}, y_{\text{Pixel}})$ coordinates. The fault counter for that pixel is then incremented. Pixels are stored only when their counter becomes non-zero, minimizing diagnostic memory usage.

### B. Super Resolution

Image SR aims to reconstruct a HR image from a LR input, recovering spatial details that may be lost during degradation (such as in aerospace domains where transmitted images may be corrupted). Early convolutional neural network approaches demonstrated the feasibility of SR. Dong *et al.* introduced SRCNN [9], which significantly outperformed traditional techniques by learning nonlinear mappings between LR and HR patches. Subsequent works such as VDSR [10] and EDSR [11] advanced performance through deeper architectures, residual learning, and improved training strategies. Attention-based models like RCAN [12] incorporated channel-attention mechanisms to enhance feature selectivity and representation capability. Generative Adversarial Networks (GANs) marked a turning point in perceptual-quality SR. Ledig *et al.* proposed SRGAN [13], which combined adversarial and perceptual losses to synthesize high-frequency textures resembling natural images. ESRGAN [14] extended this paradigm with enhanced residual blocks and a relativistic discriminator, achieving sharper and more realistic reconstructions. Despite their strengths in perceptual fidelity, GAN-based methods may introduce hallucinated artifacts, which can be problematic in domains requiring precise reconstruction.

More recently, transformer-based SR models have exploited global self-attention to capture long-range dependencies more effectively than local CNN operators. SwinIR [15], built upon the Swin Transformer architecture, demonstrated strong performance across real-image SR tasks. In parallel, diffusion-based approaches have emerged as a promising direction, offering iterative refinement capabilities that reduce hallucination artifacts while producing visually coherent textures [16].

In line with [17], we integrate the idea of SR into existing semantic segmentation pipelines, adopting and modifying the UNet [18] Neural Network (NN), as detailed in Table IV.

## III. Proposed Approach

This research proposes a novel method for reconstructing off-chip the failing memory bitmaps with high accuracy, leveraging an SR AI model.

Consider testing a portion of memory, hereafter referred to as a $tile$, with size $X \times Y$ cells, where $X$ denotes the bitline coordinate while $Y$ the wordline coordinate. During the on-chip test, instead of storing the exact coordinates information $X$ and $Y$ for each failure, cumulative information is saved for a predefined portion of memory sized $Z \times Z$, referred to as $Pixel$. $Z$ represents the number of memory cell bits that are aggregated into a single pixel. Hereinafter, we will refer to:

- **HR memory bitmap:** the tested portion of memory sized $X \times Y$, also named $tile$, saved with exact bit-wise diagnostic information;

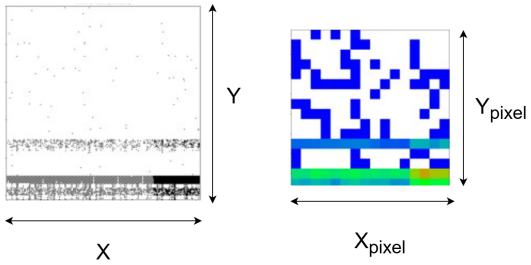


Fig. 1. Example of $X \times Y$ HR tile (on the left) and its corresponding compressed $X_{\text{pixel}} \times Y_{\text{pixel}}$ LR one (on the right). In this specific example, $X = Y = 256$ memory cells, $Z$ corresponds to 16.

- **LR memory bitmap:** the stored diagnostic information saved in a compressed format $X_{\text{pixel}} \times Y_{\text{pixel}}$ related to the tile $X \times Y$. Specifically, $X_{\text{pixel}} = \frac{X}{Z}$ and $Y_{\text{pixel}} = \frac{Y}{Z}$.

An example of HR bitmap and its corresponding LR one is illustrated in Fig. 1. The intent of this methodology is to reconstruct off-chip very accurate **predicted HR memory bitmaps** starting from compressed LR ones saved on-chip.

First, a representative dataset of fully tested, HR bitmaps is first collected at the tile level. For each HR tile, its corresponding LR Pixel representation is generated using the same $Z \times Z$ downsampling scheme implemented on-chip. These paired $(\text{LR}, \text{HR})$ samples constitute the training set for the SR reconstruction model. The dataset (in this work built starting from real RRAM faulty data in Table II, detailed in the following) is further augmented by applying spatial transformations such as flips and rotations, ensuring robustness to pattern orientation and improving generalization across diverse morphologies.

To reconstruct the HR bitmaps, we developed a dedicated architecture, named *UNetUpscale*, to perform SR on sparse and binary memory data. In *UNetUpscale*, the encoder processes the LR pixel map and extracts a hierarchy of contextual features that capture both defect density and surrounding spatial patterns. The decoder then transforms these features back into the HR domain, integrating modified upsampling layers that enforce a fixed enlargement factor of $Z$ between input and output resolutions. Skip connections between encoder and decoder stages preserve the coarse fault morphology while refining spatial detail. The network finally produces a probability map of size $X \times Y$, starting from a compressed memory bitmap of size $X_{\text{pixel}} \times Y_{\text{pixel}}$. A detailed list of introduced features is presented in Table IV.

Once the model architecture was selected, a dedicated training objective was defined to address the specific challenges of reconstructing sparse, binary fault maps. Standard pixel-wise losses are insufficient in this context, as they tend to over-penalize sparsity. To overcome these limitations, we proposed a composite loss function, referred to as *ShapeAwareLoss*, designed to jointly capture structural fidelity, textural coherence, and controlled randomness.

Let $\mathbf{P} \in [0,1]^{X \times Y}$ denote the predicted high-resolution probability map and $\mathbf{G} \in \{0,1\}^{X \times Y}$ the corresponding ground-truth binary map. The image is partitioned into non-overlapping $Z \times Z$ blocks indexed by $b \in \mathcal{B}$. We define the block mean as:

$$\mu_b = \frac{1}{|b|} \sum_{(i,j) \in b} P_{ij},$$

and a mid-density gating interval $\mathcal{M} = [\tau_{\min}, \tau_{\max}]$. A small constant $\epsilon$ is used for numerical stability.

$$\mathcal{L}_{\text{ShapeAwareLoss}} = \sum_k \lambda_k \, \mathcal{L}_k$$

The proposed *ShapeAwareLoss* is designed to guide the model toward accurate and realistic reconstructions of sparse fault maps by combining losses with complementary roles. *Region-overlap losses*, namely the soft Dice loss ($\mathcal{L}_{\text{Dice}}$) and the soft Intersection over Union (IoU) loss ($\mathcal{L}_{\text{IoU}}$), guide the model to correctly recover faulty regions in highly imbalanced data. *Shape-related losses*, including thickness consistency ($\mathcal{L}_{\text{thick}}$) and vertical Sobel alignment ($\mathcal{L}_{\text{sobel}}$), preserve the geometry of reconstructed patterns and prevent broken, overly thin, or inflated structures. To capture local spatial organization beyond pixel-wise overlap, *texture-based losses* rely on Normalized Cross-Correlation (NCC) at the block level, through the block texture term ($\mathcal{L}_{\text{block-tex}}$). This term is complemented by a gated sparse texture loss ($\mathcal{L}_{\text{sparse-tex}}$), which is applied only in mid-density regions where texture information is meaningful, and by a lightweight chess-pattern correlation ($\mathcal{L}_{\text{chess}}$), which softly encourages periodic alternation when present. At a larger scale, the *global context loss* ($\mathcal{L}_{\text{global}}$) aligns the coarse spatial distribution of faults between prediction and ground truth via Mean Squared Error (MSE).

Finally, *sparsity-aware regularization*, implemented through a false-positive penalty ($\mathcal{L}_{\text{FP}}$) and a sparse variance entropy term ($\mathcal{L}_{\text{entropy}}$), reduces background noise and preseres variability in highly sparse regions. Together, these losses enable reconstructions that remain structurally accurate across a wide range of fault patterns.

Finally, at inference time, local count coherence is enforced through a per-pixel **Top-$k$ binarization** procedure defined on the $Z \times Z$ LR map. Predicted logits are converted into binary outputs by selecting, within each $Z \times Z$ HR region, the $k$ highest-scoring pixels, where $k$ corresponds to the fault count measured in the associated LR cell. In this way, the correct number of faults is guaranteed in the HR prediction.

### A. Evaluation metrics

To evaluate the effectiveness of the framework, we adopted widely used metrics in SR and proposed a new domain-specific one. In SR, Structural Similarity Index Measure (SSIM) [19] evaluates local structural similarity and a baseline indication of visual fidelity. This is followed by the Multi-Scale Structural Similarity Index Measure (MS-SSIM) [20], which extends SSIM across multiple spatial scales and is less sensitive to small spatial shifts, resulting in more stable scores across all pattern groups. While SSIM and MS-SSIM are widely used in image SR, they were originally designed for natural

TABLE I
SHAPEAWARELOSS COMPONENTS, OPERATING SCALE, AND MATHEMATICAL FORMULATION.

| Loss Component | Operation | Mathematical formula | Scale | Role |
|---|---|---|---|---|
| Dice | Soft region overlap | $\mathcal{L}_{\text{dice}} = 1 - \frac{2\,\mathbf{PG}}{\|\mathbf{P}\|_1 + \|\mathbf{G}\|_1}$ | Pixel | Robust overlap under imbalance |
| IoU | Soft IoU | $\mathcal{L}_{\text{IoU}} = 1 - \frac{\mathbf{PG}}{\|\mathbf{P}\|_1 + \|\mathbf{G}\|_1 - \mathbf{PG}}$ | Pixel | False Positive/False Negative balancing |
| Thickness | $3\times3$ max-pool difference | $\mathcal{L}_{\text{thick}} = \|\text{MP}_{3\times3}(\mathbf{P}) - \text{MP}_{3\times3}(\mathbf{G})\|_1$ | Pixel | Width preservation |
| Vertical Sobel | Edge alignment | $\mathcal{L}_{\text{sobel}} = \|\mathbf{P} * S_y - \mathbf{G} * S_y\|_1$ | Pixel | Vertical continuity |
| Block texture | NCC on $Z\times Z$ blocks | $\mathcal{L}_{\text{block}} = 1 - \text{NCC}(\mathbf{P}_b, \mathbf{G}_b)$ | Block | Local texture similarity |
| Sparse texture | Gated NCC (mid-density) | $\mathcal{L}_{\text{sparse}} = \mathbb{1}_{\mu_b \in \mathcal{M}}(1 - \text{NCC}(\mathbf{P}_b, \mathbf{G}_b))$ | Block | Stabilize mid-density patterns |
| Canonical chess | Template NCC | $\mathcal{L}_{\text{chess}} = \mathbb{1}_{\mu_b \in \mathcal{M}}(1 - \text{NCC}(\mathbf{P}_b, \mathbf{C}))$ | Block | Soft periodic bias |
| Global context | Pooled MSE | $\mathcal{L}_{\text{global}} = \|\text{Pool}(\mathbf{P}) - \text{Pool}(\mathbf{G})\|_2^2$ | Global | Low-frequency alignment |
| False positive | Background mean | $\mathcal{L}_{\text{FP}} = \mathbb{E}_{G_{ij}=0}[P_{ij}]$ | Pixel | Noise suppression |
| Sparse variance entropy | Entropy and variance | $\mathcal{L}_{\text{entropy}} = -H(\mu_b) - \text{Var}(\mu_b)$ | Block | Preserve dispersion |

images with continuous intensity variations. In our problem, the data consist of binary and highly sparse faulty bitmaps, characterized by thin lines, crossings, grid-like patterns, and sparse points. Under these conditions, standard SSIM-based metrics tend to over-penalize small spatial shifts and are highly sensitive to minor misalignments in thin structures. To address these issues, this work introduces a composite metric: the **Binary-SSIM**. Let $B_p, B_{gt} \in \{0,1\}^{H\times W}$ denote the predicted and ground-truth binary fault maps of spatial resolution $H \times W$. To better capture task-specific structural fidelity, we define *Binary-SSIM* as a weighted combination of SSIM scores computed on multiple transformed representations of these maps:

$$\begin{aligned}\text{Binary-SSIM}(B_p, B_{gt}) = {} & w_b\,\text{SSIM}(\Phi_b(B_p), \Phi_b(B_{gt})) \\ & + w_e\,\text{SSIM}(\Phi_e(B_p), \Phi_e(B_{gt})) \\ & + w_d\,\text{SSIM}(\Phi_d(B_p), \Phi_d(B_{gt}))\,,\end{aligned} \tag{3}$$

where $\Phi_b(\cdot)$, $\Phi_e(\cdot)$, and $\Phi_d(\cdot)$ denote blur, edge, and density transformations, and the weights satisfy $w_b + w_e + w_d = 1$.

- Blur SSIM: *Is the overall shape and extent of the structure correctly recovered, even in the presence of small spatial shifts?* The blur transformation $\Phi_b(\cdot)$ focuses on coarse structure by smoothing out fine details. It is computed by convolving the binary map with Gaussian kernels at multiple scales: $\Phi_b(B) = \{G_\sigma * B \mid \sigma \in \{1, 2, 4\}\}$, where $G_\sigma$ is a Gaussian kernel with standard deviation $\sigma$, and $*$ denotes convolution. SSIM is computed at each scale and then averaged. This term rewards agreement in overall shape and extent, and it is tolerant to small spatial shifts.
- Edge SSIM: *Are orientations, edges, and structural continuities preserved?* The edge transformation $\Phi_e(\cdot)$ checks whether structural boundaries are preserved. It is computed as the gradient magnitude using Sobel filters:

$$\Phi_e(B) = \sqrt{(S_x * B)^2 + (S_y * B)^2}, \tag{4}$$

where $S_x$ and $S_y$ are the horizontal and vertical Sobel filters, respectively. Applying SSIM to these gradient-magnitude maps emphasizes correct edge orientation, continuity, and location.
- Density SSIM: *Is the number of faults in each region correct?* The density transformation $\Phi_d(\cdot)$ compares the coarse spatial distribution of faults. It is obtained by applying average pooling to the binary map: $\Phi_d(B) = \text{AvgPool}(B; k = 16, s = 8)$, where $k$ is the pooling kernel size and $s$ is the stride. Each pooled value corresponds to the fraction of failing cells in a local region. Applying SSIM to these density maps evaluates whether faults are distributed in the correct areas while being insensitive to small local rearrangements.

## IV. EXPERIMENTAL RESULTS

The effectiveness of the proposed framework is validated using failing RRAM bitmaps from Infineon's AURIX™ TC4x family of microcontrollers [8].

With Z=16, each 256×256 memory bitmap (tile) is associated with a 16×16 LR grid, where each LR pixel records the number of faulty cells in a corresponding 16×16 HR region. Each tile can contain a shape from a different set; depending on the number of faults in each tile ($f_{\text{count}}$ as detailed in Table II), we labeled them as sparse, cluster, dense, or linear. Specifically, a tile is classified as linear when the ratio between the largest and smallest eigenvalues ($\lambda_{\text{max}}$ and $\lambda_{\text{min}}$) of the fault-pixel covariance matrix exceeds 10, indicating a strongly elongated, line-like spatial distribution. The reader should notice that the intent of the framework was not to classify shapes (sparse, dense, cluster, and linear), but they have been only used to prepare the dataset; the final prediction is evaluated as a segmentation and super-resolution task.

From failing RRAM memories, we collected the following faulty bitmap information (Table II), which we split into 70% - 15% - 15% partitions for the training, validation and test, respectively.

TABLE II
RRAM FAILING BITMAP ANALYSIS AND CATEGORIES

| Dataset | Fault Count ($f_{count}$) | Train | Validation | Test |
|---|---|---|---|---|
| Sparse | $f_{\text{count}} \leq 100$ | 26,660 | 5,713 | 5,714 |
| Cluster | $100 \leq f_{\text{count}} \leq 1000$ | 725 | 155 | 157 |
| Dense | $f_{\text{count}} > 1000$ | 1,552 | 332 | 334 |
| Linear | $\frac{\lambda_{\max}}{\lambda_{\min}} \geq 10$ | 9,277 | 1,987 | 1,987 |
| Total | | 38,214 | 8,187 | 8,192 |

TABLE III
PROPOSED UNETUPSCALE ARCHITECTURE

| Layer Name | Layer Type | Output size (C × H × W) |
|---|---|---|
| input | - | 1 × 16 × 16 |
| enc1.0 | Double Conv 5×5 + BN + ReLU + Dropout + SE | 32 × 16 × 16 |
| pool1 | Downsampling (maxpool 2×2) | 32 × 8 × 8 |
| enc2.0 | Double conv 5×5 + BN + ReLU + Dropout + SE | 64 × 8 × 8 |
| pool2 | Downsampling (maxpool 2×2) | 64 × 4 × 4 |
| enc3.0 | Double conv 5×5 + BN + ReLU + Dropout + SE | 128 × 4 × 4 |
| prebottleneck.pool | Downsampling (maxpool 2×2) | 128 × 2 × 2 |
| bottleneck | parallel dilated 3×3 (rates 1,2,4) → sum + BN + ReLU; spatial attention (7×7) | 256 × 2 × 2 |
| dec3.0 | Upsampling (bilinear ×2) + conv 3×3 (256→128) | 128 × 4 × 4 |
| dec3.1 | Double conv 5×5 + BN + ReLU + Dropout + SE (with skip from enc3) | 128 × 4 × 4 |
| dec2.0 | Upsampling (bilinear ×2) + conv 3×3 (128→64) | 64 × 8 × 8 |
| dec2.1 | Double conv 5×5 + BN + ReLU + Dropout + SE (with skip from enc2) | 64 × 8 × 8 |
| dec1.0 | Upsampling (bilinear ×2) + conv 3×3 (64→32) | 32 × 16 × 16 |
| dec1.1 | Double conv 5×5 + BN + ReLU + Dropout + SE (with skip from enc1) | 32 × 16 × 16 |
| head | Transposed Convolution (kernel=16, stride=16) | 1 × 256 × 256 |

Details of the *UNetUpscale* model architecture are given in Table III. This model was trained to predict a $256 \times 256$ faulty bitmap from a $16 \times 16$ LR input. Training used the Adam optimizer with learning rate $1 \times 10^{-4}$ and weight decay $1 \times 10^{-3}$, together with a *ReduceLROnPlateau* scheduler (factor $0.5$, patience 3). The model was trained for 60 epochs with batch size 64 and dropout 0.1. Automatic mixed precision was enabled, tensors used the `channels_last` format, cuDNN benchmarking was active, and gradients were clipped to a maximum norm of 1.0. The overall loss (ShapeAwareLoss) is defined as a weighted sum of the individual components, using fixed weights empirically tuned to ensure stable training; the specific weights assigned to each loss term are: $0.35$ for $\mathcal{L}_{\text{Dice}}$, $0.30$ for $\mathcal{L}_{\text{IoU}}$, $0.10$ for $\mathcal{L}_{\text{thick}}$, $0.15$ for $\mathcal{L}_{\text{sobel}}$, $0.10$ for $\mathcal{L}_{\text{block-tex}}$, $0.15$ for $\mathcal{L}_{\text{sparse-tex}}$, $0.05$ for $\mathcal{L}_{\text{chess}}$, $0.15$ for $\mathcal{L}_{\text{global}}$, $0.05$ for $\mathcal{L}_{\text{FP}}$, and $0.20$ for $\mathcal{L}_{\text{entropy}}$.

For evaluation and downstream use, logits are converted to binary maps via per-pixel top-k binarization. Within each 16×16 HR region, we select the k highest-scoring positions, where k equals the LR faults count for that region. This guarantees that the total number of faults for each region is correct, while still letting the NN model decide how values are distributed within each region based on what it has learned.

To quantify how training on a single pattern affects behavior on the entire test set, we train separate NN models using only one label category at a time (only clustered, only linear, only sparse, or only dense), while keeping the validation/test protocol unchanged. Table V reports the evaluation results across pattern-specific subsets (Sparse, Clustered, Dense, and Linear) and on the complete dataset. Results on the test dataset

TABLE IV
COMPARISON BETWEEN STANDARD U-NET AND PROPOSED UNETUPSCALE: MAIN DIFFERENCES.

| Design Aspect | Standard U-Net [18] for Image Segmentation | Proposed UNetUpScale for Super Resolution |
|---|---|---|
| Input/Output Size | Input and output have the same spatial size (segmentation). | Uses a $16 \times 16$ LR Pixel map as input and produces a $256 \times 256$ HR bitmap, performing a fixed $\times 16$ super-resolution. |
| Upsampling Strategy | Transposed convolutions at each decoder stage. | Decoder uses bilinear upsampling followed by $3 \times 3$ convolution for refinement; final resolution jump performed by a single transposed convolution (kernel 16, stride 16). |
| Convolution Kernels | $3 \times 3$ convolutions throughout. | All encoder and decoder blocks use $5 \times 5$ convolutions to increase receptive field and better interpret aggregated Pixel information. |
| Feature Recalibration | No channel-wise attention modules. | Each block includes a Squeeze-and-Excitation (SE) module to emphasize informative feature channels and suppress irrelevant ones. |
| Regularization | Dropout rarely used. | Dropout is applied consistently across encoder and decoder blocks to avoid overfitting on sparse binary fault patterns. |
| Bottleneck Structure | Single $3 \times 3$ convolution blocks. | The bottleneck uses three parallel dilated convolutions (rates 1, 2, 4), whose outputs are summed and passed through BN, ReLU, and a $7 \times 7$ spatial attention module for multi-scale context extraction. |

TABLE V
EFFECTIVENESS OF THE APPROACH IN RECONSTRUCTING MEMORY-FAILING BITMAPS WITH STRUCTURAL FIDELITY.

| Metric (%) | Only Sparse | Only Clustered | Only Dense | Only Linear | Full Dataset |
|---|---|---|---|---|---|
| SSIM | 96.49 | 96.63 | 96.56 | 97.16 | 97.36 |
| MS-SSIM | 96.84 | 96.64 | 96.92 | 97.37 | 97.62 |
| Binary-SSIM | 97.67 | 97.60 | 97.95 | 98.24 | **98.40** |

show that the method is capable of reconstructing the failing bitmaps with very high SSIM (above 97%), with the proposed Binary-SSIM achieving the best results (98.40%).

To conclude, we report time measurements. Timing results are presented for: *single-tile latency*, i.e., the time to process one LR tile (16×16) to produce a 256×256 logits map and per-pixel top-k binarization, reporting **single tile total (ms)**; and *full-chip throughput*, the time required to process a configurable number of tiles (default 368, matching a 5,888×4,224 array tiled at 256×256) in batches, reporting **full chip total (s)**. All experiments were conducted using PyTorch 2.7.0+cu126 on an NVIDIA L40 GPU with CUDA support enabled.

Table VI analyzes throughput as a function of batch size. While the latency of a single tile remains largely unaffected by batch size, the time required to process the full chip exhibits a clear dependence on batching, with larger batches improving GPU utilization and reducing total runtime. In our setup, batch

TABLE VI
THROUGHPUT IMPROVES MARKEDLY AS BATCH INCREASES; BEST AT BATCH=128 ( 0.298 MS/TILE) ON THE L40.

| Tiles | Batch size | Batches | Single tile (ms) | Full per_tile (ms) | Full chip (s) |
|---|---|---|---|---|---|
| 368 | 16 | 23 | 3.372 | 0.358 | 0.132 |
| 368 | 32 | 12 | 3.458 | 0.461 | 0.17 |
| 368 | 64 | 6 | 3.382 | 0.41 | 0.151 |
| 368 | 128 | 3 | 3.404 | 0.298 | 0.11 |
| 1472 | 64 | 23 | 3.47 | 0.152 | 0.223 |

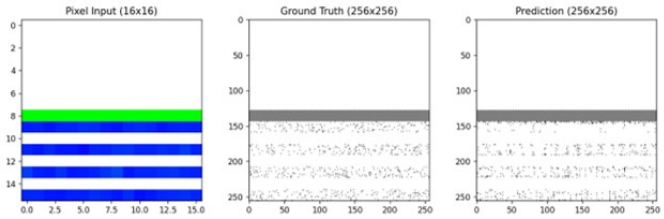


Fig. 2. *UNetUpScale* prediction of a faulty RRAM bitmap portion.

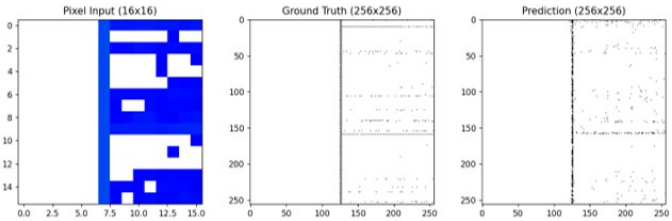


Fig. 3. *UNetUpScale* prediction of a faulty RRAM bitmap portion.

sizes in the range of 64–128 provide the best trade-off between throughput and batching overhead, achieving the lowest per-tile execution time for full-chip inference. Finally, with a batch size of 64, we executed a long run (last row of Table VI), and figured out that when the GPU is kept busy across many batches, the full per-tile time drops further (0.152 ms/tile), reflecting a very good throughput.

Examples of reconstructed faulty bitmaps with the *UNetUpScale* model are provided in Fig. 2 and Fig. 3. The developed model takes as input the encoded 16×16 pixel image (the information saved in the on-chip memory buffer at the end of the test), shown on the leftmost part of Fig. 2 and Fig. 3. Then, it predicts the corresponding 256×256 faulty bitmaps (figures on the right). The ground truth, shown in the middle of the figures, is given only for visual comparison. Therefore, the proposed method demonstrates strong effectiveness in macroscopic inspection.

## V. Conclusions

To conclude, the proposed framework offers a practical solution to reconstruct off-chip, with very high accuracy and fidelity, faulty RRAM bitmaps, starting from the lossy pixel-based encoding [3]. The adoption of the lossy encoding is necessary to avoid on-chip memory buffer saturation (which may occur if all exact coordinates for faulty bits are saved). The next activities will focus on improving the pixel-wise prediction accuracies by integrating precise information and combining lossless and lossy encoding to yield better results.

## References

[1] P. Bernardi, G. Insinga, G. Paganini, R. Cantoro, P. Beer, M. Coppetta, N. Mautone, G. Carnevale, P. Scaramuzza, and R. Ullmann, "Optimized diagnostic strategy for embedded memories of automotive systems-on-chip," in *2022 IEEE European Test Symposium (ETS)*, 2022, pp. 1–6.

[2] P. Bernardi, B. Borio, G. Insinga, B. Mendicino, M. Battilana, M. Coppetta, N. Mautone, P. Scaramuzza, F. Tengler, and R. Ullmann, "Late breaking results: A data compaction strategy for extensive test flows of memories embedded in automotive socs," in *2025 Design, Automation Test in Europe Conference (DATE)*, 2025, pp. 1–2.

[3] G. Insinga, M. Battilana, M. Coppetta, N. Mautone, G. Carnevale, M. Giltrelli, P. Scaramuzza, and R. Ullmann, "Density-oriented diagnostic data compression strategy for characterization of embedded memories in automotive systems-on-chip," in *2023 IEEE European Test Symposium (ETS)*, 2023, pp. 1–6.

[4] Z. Wang, J. Chen, and S. C. H. Hoi, "Deep learning for image super-resolution: A survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, no. 10, pp. 3365–3387, 2021.

[5] Z. Deng, Z. Chen, S. Niu, T. H. Li, B. Zhuang, and M. Tan, "Efficient test-time adaptation for super-resolution with second-order degradation and reconstruction," in *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. [Online]. Available: https://openreview.net/forum?id=IZRlMABK4l

[6] S. Lei, Z. Shi, and Z. Zou, "Super-resolution for remote sensing images via local–global combined network," *IEEE Geoscience and Remote Sensing Letters*, vol. 14, no. 8, pp. 1243–1247, 2017.

[7] J. Li *et al.*, "A systematic survey of deep learning-based single-image super-resolution," *ACM Comput. Surv.*, vol. 56, no. 10, May 2024. [Online]. Available: https://doi.org/10.1145/3659100

[8] Infineon Technologies AG, "32-bit tricore™ aurix™ – tc4x," https://www.infineon.com/products/microcontroller/32-bit-tricore/aurix-tc4x, Infineon Technologies AG, 2025, accessed: 2026-01-07.

[9] C. Dong, C. C. Loy, K. He, and X. Tang, "Image super-resolution using deep convolutional networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 2, pp. 295–307, 2016.

[10] J. Kim, J. K. Lee, and K. M. Lee, "Accurate image super-resolution using very deep convolutional networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 1646–1654.

[11] B. Lim, S. Son, H. Kim, S. Nah, and K. M. Lee, "Enhanced deep residual networks for single image super-resolution," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2017, pp. 1132–1140.

[12] Y. Zhang *et al.*, "Image super-resolution using very deep residual channel attention networks," in *Proceedings of the European Conference on Computer Vision*, 2018, pp. 294–310.

[13] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Aitken, A. Tejani, J. Totz, Z. Wang, and W. Shi, "Photo-realistic single image super-resolution using a generative adversarial network," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 4681–4690.

[14] X. Wang *et al.*, "Esrgan: Enhanced super-resolution generative adversarial networks," in *Proceedings of the European Conference on Computer Vision Workshops*, 2018, pp. 63–79.

[15] J. Liang, J. Cao, G. Sun, K. Zhang, L. Van Gool, and R. Timofte, "Swinir: Image restoration using swin transformer," in *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, 2021, pp. 1833–1844.

[16] C. Saharia, J. Ho, W. Chan, T. Salimans, D. J. Fleet, and M. Norouzi, "Image super-resolution via iterative refinement," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 2, pp. 1413–1428, 2023.

[17] L. Wang, D. Li, Y. Zhu, L. Tian, and Y. Shan, "Dual super-resolution learning for semantic segmentation," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 3773–3782.

[18] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, N. Navab, J. Hornegger, W. M. Wells, and A. F. Frangi, Eds. Cham: Springer International Publishing, 2015, pp. 234–241.

[19] Z. Wang, A. Bovik, H. Sheikh, and E. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, 2004.

[20] Z. Wang, E. Simoncelli, and A. Bovik, "Multiscale structural similarity for image quality assessment," in *The Thrity-Seventh Asilomar Conference on Signals, Systems Computers, 2003*, vol. 2, 2003, pp. 1398–1402 Vol.2.