LoDiSS: Learning in Low-Dimensional Seeded Subspaces for Long-Term Test-Time Recovery of Floating-Point Neural Weights

Anonymous authors
Paper under double-blind review

Abstract

As machine learning models are increasingly deployed on standalone edge devices, their weights are exposed to random and adversarial bit flips that can accumulate over time. Most defenses target integer weights or tolerate corruption without repair, leaving floating-point deployment insufficiently protected. Prior real-valued weight recovery suffers from inefficient training and recovery due to redundant variables and group-wide optimization, while inadequate handling of rounding and nonfinite values can lead to unreliable recovery. We introduce LoDiSS, which learns equivalent recoverable weights in low-dimensional seeded subspaces with less training state. Shared pseudorandom transformations efficiently regenerate recovery constraints at test time, while calibrated iterative recovery handles rounding and nonfinite values and reconstructs only suspected weights, without training data or a clean model copy. Across 20 classification settings, LoDiSS retains approximately 100% of clean accuracy under repeated random faults and outperforms baselines under adversarial and recovery-aware adaptive attacks, with mean training and recovery speedups of $2.11\times$ and $17.59\times$ over the existing baseline. Together with evaluations across additional tasks, these results demonstrate reliable, efficient, and task-agnostic floating-point recovery for long-term standalone deployment.

1 Introduction

As machine learning models are increasingly deployed on standalone edge devices, their stored weights are exposed to random bit flips caused by factors such as reduced supply voltage, temperature variations, radiation exposure, and device aging (Chang et al., 2017; Liu et al., 2013; Rech, 2024; Gebregiorgis et al., 2016). These deployments can also give adversaries physical access to devices, allowing them to deliberately flip a small number of stored weight bits and disrupt model behavior (Hong et al., 2019). When maintenance and access to clean model copies are limited, corrupted weights can persist in memory, allowing subsequent faults to accumulate and further degrade inference. Existing defenses address these threats through fault tolerance, which aims to preserve accurate predictions despite corruption, and fault recovery, which repairs the stored weights themselves. Despite this progress, several limitations remain for reliable long-term deployment.

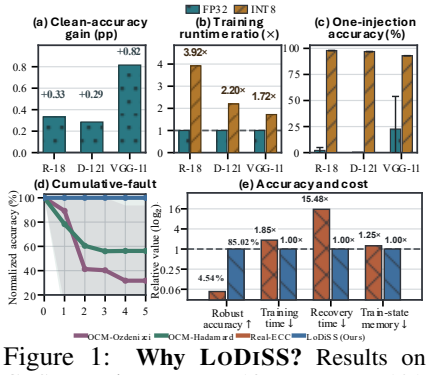


Figure 1: **Why LoDiSS?** Results on GTSRB using ResNet-18, DenseNet-121, and VGG-11 illustrate floating-point benefits (a,b), vulnerability to single and repeated faults (c,d), and recovery accuracy and observed runtime gains over Real-ECC (e).

First, **most existing studies target integer-quantized weights, leaving a gap in robust floating-point deployment**. Although edge deployments use both floating-point and integer-quantized models, prominent defenses focus on integer-quantized weights through bit-flip-aware training, learned quantization ranges, and in-place ECC (Stutz et al., 2021; Chitsaz et al., 2023; Guan et al., 2019). These methods have been neither demonstrated nor established for floating-point weight protection. Their designs exploit bounded integer ranges or integer-specific bit layouts, whereas floating-point exponent faults can produce extreme magnitudes or nonfinite values that require different treatment.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

This leaves a practical gap for deployments where floating-point representations offer advantages over integer quantization. Retaining floating-point weights avoids additional integer quantization error and the associated calibration or fine-tuning, while supporting native CPU/GPU execution (Kuzmin et al., 2022; Esser et al., 2020; Cai et al., 2023). Depending on the implementation, it can also avoid conversion overhead that offsets the computational gains of integer inference (Xiao et al., 2023). Moreover, conventional FP16/FP32 storage does not require shared dequantization scales, whose corruption can affect an entire weight group (Wang et al., 2025). In the evaluated comparisons, floating-point models achieve higher clean accuracy and shorter observed training time, but unprotected models suffer greater degradation at the same per-bit fault rate (Figure 1a–c). These advantages and vulnerabilities reinforce the need for protection tailored to floating-point weights.

Second, **fault tolerance can eventually fail as bit-flip faults accumulate**. In practical deployments, corrupted weights can remain in memory, allowing subsequent faults to add to existing damage. A model that initially maintains accurate predictions may therefore lose robustness over time. Output code matching (OCM)-based methods offer a potential route to floating-point fault tolerance by introducing redundancy through class codewords without requiring integer weights (Özdenizci & Legenstein, 2022; Yu et al., 2024). Most critically, these methods aim to preserve predictions without repairing corrupted weights, allowing damage to persist and accumulate, while their restriction to classification and previously untested effectiveness under floating-point bit flips introduce additional limitations. Figure 1d illustrates how OCM accuracy declines as faults accumulate across repeated injections. Reliable long-term deployment of floating-point models can therefore benefit from weight recovery rather than fault tolerance alone.

Third, however, **existing real-valued recovery beyond integer-quantized models remains insufficiently adapted to neural deployment**. The recently proposed Real-ECC (Li et al., 2026) offers a promising direction for neural weight recovery by adapting linear constraints from signal processing and using linear programming to estimate sparse errors (Candes & Tao, 2005; Candes & Randall, 2008). Although this formulation enables numerical recovery without relying on integer-specific representations, its direct adaptation leaves three practical limitations. *i) Inefficient training and recovery.* Training retains full-dimensional variables despite fewer independent degrees of freedom, while recovery solves for corrections across an entire weight group. These choices retain redundant training state and add recovery overhead, a critical concern on edge devices where repairs compete with inference for limited computational resources. *ii) Costly test-time regeneration.* Rebuilding the recovery basis through Real-ECC's QR-based construction requires generating and factorizing a dense matrix, adding computational overhead before weight repair. This cost is particularly consequential on resource-constrained edge devices. *iii) Limited reliability under floating-point faults.* Exact consistency can mistake ordinary rounding for corruption and modify otherwise clean weights, while nonfinite values caused by bit flips are not handled explicitly and can prevent recovery. Its evaluation also covers few models and datasets without comparisons against competing protection methods, leaving its practical effectiveness insufficiently established. The results in Figure 1e highlight Real-ECC's limited post-recovery accuracy and high observed training and recovery costs.

These limitations motivate a central question: ***"Can we sustain reliable floating-point inference under persistent bit flips through efficient training and test-time weight recovery?"*** We address this question with LoDiSS, which reinterprets linear constraints from signal processing as *low-dimensional seeded subspaces for learning recoverable floating-point weights*. A seeded pseudorandom number generator (PRNG) fixes the linear transformations defining each layer's subspace and its recovery constraints before training, then reproduces them at test time. Direct subspace learning preserves the expressiveness of an equivalent full-dimensional projection while enabling efficient training. The regenerated constraints are shared across weight rows and reused across repairs, reducing generation overhead and making recovery more practical on resource-constrained edge devices. We calibrate iterative recovery to floating-point rounding and handle nonfinite values explicitly to improve repair reliability. Selective reconstruction further reduces test-time repair cost and preserves unselected weights. Together, these designs enable task-agnostic recovery of floating-point weights without training data or a clean model copy.

Across four classification datasets and five architectures, LoDiSS retains approximately 100% of clean accuracy under long-term random faults and achieves the highest mean robust accuracy in all 20 adversarial and adaptive-attack settings. Compared with Real-ECC, observed mean training and recovery speedups are $2.11\times$ and $17.59\times$, with approximately 20% less weight-related training state. Results on time-series classification, reconstruction, language modeling, and detection support

task-agnostic bit-flip recovery, while further evaluations demonstrate robustness across floating-point precisions and subspace dimensions.

2 THREAT MODELS

We consider persistent bit flips in stored floating-point convolutional and linear weights, including embedding weights for language modeling. Recovery uses pseudorandom values regenerated at test time and a single retained global calibration threshold, without requiring training data, a clean weight copy, or prior knowledge of fault locations.

Random bit-flip threat. Let N_{bit} denote the number of eligible bits, $X \in \{0,1\}^{N_{\text{bit}}}$ the current stored weight state, and $M \in \{0,1\}^{N_{\text{bit}}}$ a fault mask. Applying the mask gives $X \oplus M$, where $\oplus$ denotes bitwise exclusive OR and toggles positions with $M_i = 1$. Following Stutz et al. (2021), each eligible bit is independently toggled with probability p, the bit-error rate (BER) per injection. At injection t, the random mask satisfies $M_i^{(t)} \sim \text{Bernoulli}(p)$ independently across bits and injections, producing $N_{\text{bit}}p$ flips in expectation.

Adversarial bit-flip threat. We consider a white-box *bit-flip attack* (BFA) based on progressive bit search (Rakin et al., 2019). The adversary accesses model weights, gradients, and a small labeled calibration subset $\mathcal{C}$ to select damaging weight-bit flips under a fixed budget B. Let $\mathcal{L}_{\mathcal{C}}(X)$ denote the model's mean prediction loss on $\mathcal{C}$ using weight state X. At each search step, gradients identify promising candidates, and their actual prediction losses determine the next flip. Selected flips accumulate at distinct bit positions. BFA targets loss before recovery without querying the recovery procedure.

Adaptive adversarial bit-flip threat. We consider a stronger adversary who knows the recovery mechanism and selects faults to maximize prediction loss after recovery. Under the same budget B of distinct bit flips, adaptive BFA optimizes

$$\max_{\substack{M \in \{0,1\}^{N_{\text{bit}}} \\ \|M\|_0 \le B}} \mathcal{L}_{\mathcal{C}}(\mathsf{Repair}(X \oplus M)), \tag{1}$$

where $\|M\|_0$ counts flipped bits and Repair returns the recovered weight state. Bounded greedy search uses an identity-gradient approximation through numerical recovery to propose candidates (Athalye et al., 2018), then ranks them by executing actual recovery on temporary model copies, including candidates producing nonfinite weights. Appendix C.10 formalizes this search.

Long-term threat. Let $X^{(0)}$ denote the clean weight bits. At interval t, a random or adversarial mask $M^{(t)}$ changes the preceding stored state, followed by recovery:

$$\widetilde{X}^{(t)} = X^{(t-1)} \oplus M^{(t)}, \qquad X^{(t)} = \mathsf{Repair}\Big(\widetilde{X}^{(t)}\Big), \tag{2}$$

where Repair is the identity for methods without recovery. Each interval may contain multiple flips. Subsequent faults act on the actual repaired weights, retaining any reconstruction error without resetting to the clean model. In contrast, fixed-budget attacks accumulate flips without committing repairs; recovery is applied only to copies used for evaluation.

3 METHOD

3.1 PRELIMINARIES

Frames and consistency checks in signal processing. In signal processing, a *frame* represents a signal through redundant linear measurements. Let $1 \le d < D$ be integers, $\boldsymbol{x} \in \mathbb{R}^d$, and $\{\varphi_i\}_{i=1}^D$ span $\mathbb{R}^d$. The analysis operator $F \in \mathbb{R}^{D\times d}$ stacks $\varphi_i^\top$ as rows. A *Parseval frame* satisfies

$$\boldsymbol{\alpha} = F\boldsymbol{x} \in \mathbb{R}^D, \qquad F^\top F = \mathrm{I}_d, \tag{3}$$

where $\boldsymbol{\alpha}$ contains the frame coefficients and I_d is the identity matrix. Its synthesis operator recovers $\boldsymbol{x} = F^\top \boldsymbol{\alpha}$, and $\|\boldsymbol{\alpha}\|_2 = \|\boldsymbol{x}\|_2$ (Casazza et al., 2011). Let $P = FF^\top$ project onto the column space of F and $N = \mathrm{I}_D - P$ onto its orthogonal complement. For an error $\boldsymbol{\epsilon} \in \mathbb{R}^D$ and observation $\widetilde{\boldsymbol{\alpha}} = \boldsymbol{\alpha} + \boldsymbol{\epsilon}$, the *syndrome residual*

$$r(\widetilde{\boldsymbol{\alpha}}) = N\widetilde{\boldsymbol{\alpha}} = \widetilde{\boldsymbol{\alpha}} - F(F^\top \widetilde{\boldsymbol{\alpha}}) = N\boldsymbol{\epsilon} \tag{4}$$

removes the unknown signal. Completing F to an orthogonal basis gives a check matrix $H \in \mathbb{R}^{(D-d)\times D}$ whose rows span the orthogonal complement of the column space of F. Since $HF = 0$ and $H^\top H = N$, the conventional syndrome $H\widetilde{\boldsymbol{\alpha}}$ carries the same information (Candes & Tao, 2005; Candes & Randall, 2008). The residual requires only F and $F^\top$; direct projection alone retains $P\boldsymbol{\epsilon}$. Vector norms $\|\cdot\|_1$, $\|\cdot\|_2$, and $\|\cdot\|_\infty$ denote the sum of absolute entries, Euclidean norm, and largest absolute entry; the matrix norm $\|\cdot\|_2$ is spectral.

From signal recovery to neural weight recovery. Real-ECC (Li et al., 2026) connects signal recovery to neural weights by enforcing linear relationships among their values. To describe this adaptation, let $f(\cdot; \{W_\ell\}_{\ell=1}^L, \psi)$ denote a neural network with L weight layers selected for recovery and remaining parameters ψ. Layer $f_\ell(\cdot; W_\ell)$ has flattened weights $W_\ell \in \mathbb{R}^{n_\ell \times D_\ell}$, with n_ℓ outputs and D_ℓ weights per output, including input-channel and kernel positions for a convolution. Define row $w_{\ell,a} = (W_\ell)_{a,:}^\top$ for $a = 1, \ldots, n_\ell$. Real-ECC obtains orthonormal checks $H_\ell \in \mathbb{R}^{(D_\ell - d_\ell)\times D_\ell}$ through QR factorization, leaving d_ℓ independent degrees of freedom per group, where $1 \le d_\ell < D_\ell$. A seed reproduces the random constraint inputs, and repeating QR under fixed numerical conventions regenerates the checks. For comparison with LoDiSS, we apply the same checks to each output row within a layer, although Real-ECC also permits groups spanning multiple rows. Define $P_\ell = \mathrm{I}_{D_\ell} - H_\ell^\top H_\ell$ and $N_\ell = H_\ell^\top H_\ell$. Training projects a full-dimensional variable V_ℓ, while recovery estimates a sparse correction u for corrupted weights $y = w_{\ell,a} + e$, with $e \in \mathbb{R}^{D_\ell}$:

$$W_\ell = V_\ell P_\ell, \qquad V_\ell \in \mathbb{R}^{n_\ell \times D_\ell}, \tag{5}$$

$$\widehat{e} \in \arg\min_{u \in \mathbb{R}^{D_\ell}} \|u\|_1 \quad \text{subject to} \quad N_\ell(y - u) = 0. \tag{6}$$

In Equation 6, the ℓ_1 objective favors sparse corrections, while the constraint requires the corrected weights to satisfy the linear checks. Introducing auxiliary variables for the absolute values expresses this recovery problem as a *linear program* (LP). While this formulation establishes a promising connection between signal recovery and neural weight learning, three challenges remain for practical deployment.

Challenge 1: Inefficient training and recovery. V_ℓ retains $n_\ell D_\ell$ variables for $n_\ell d_\ell$ degrees of freedom, while recovery optimizes corrections across an entire group. This retains redundant optimization state and can increase runtime, especially when edge-device repairs compete with inference.

Challenge 2: Costly test-time regeneration of recovery constraints. A seed reproduces the random inputs but does not eliminate dense QR factorization when discarded constraints are rebuilt. This initialization adds runtime before repair, making inexpensive regeneration important for repeated recovery on edge devices.

Challenge 3: Unreliable recovery of stored floating-point weights. Floating-point rounding can make clean stored weights violate the exact constraints in Equation 6. Recovery may therefore modify weights that have not been corrupted. Exponent faults can also produce extreme or nonfinite values, requiring precision-aware tolerances and explicit handling of invalid values.

3.2 Proposed Method: LoDiSS

Learning recoverable weights in seeded subspaces. To address Challenge 1, LoDiSS interprets the checks as *learning in low-dimensional seeded weight subspaces*. A seeded PRNG directly constructs a fixed Parseval frame $F_\ell \in \mathbb{R}^{D_\ell \times d_\ell}$ without forming H_ℓ. Its column space $\mathcal{S}_\ell = \mathrm{range}(F_\ell)$ is the layer's *seeded subspace*, reproducible at test time independently of learned weights. Under matched constraints, $H_\ell F_\ell = 0$, $P_\ell = F_\ell F_\ell^\top$, and $N_\ell = \mathrm{I}_{D_\ell} - P_\ell$. Writing $\varphi_{\ell,i}^\top$ for frame row i, we learn $Z_\ell \in \mathbb{R}^{n_\ell \times d_\ell}$ instead of V_ℓ (Li et al., 2018):

$$W_\ell = Z_\ell F_\ell^\top, \qquad Z_\ell^{(0)} = \sqrt{D_\ell / d_\ell}\, W_\ell^{\mathrm{init}} F_\ell. \tag{7}$$

Thus the task loss learns signal representations whose expanded coefficients are neural weights. The ratio d_ℓ / D_ℓ controls redundancy. This preserves the admissible weights of the ambient projection with less training state (Appendix C.1, Appendix C.2); initialization from W_ℓ^{init} preserves average variance for a source isotropic conditional on F_ℓ. For dataset $\mathcal{D}$, minibatch $\mathcal{B}$, and task loss $\mathcal{L}$, training and backpropagation are

$$\min_{\{Z_\ell\}_{\ell=1}^L, \psi} \mathbb{E}_{\mathcal{B}\sim\mathcal{D}} \mathcal{L}\big(\{Z_\ell F_\ell^\top\}_{\ell=1}^L, \psi; \mathcal{B}\big), \tag{8}$$

$$\nabla_{Z_\ell} \mathcal{L} = (\nabla_{W_\ell} \mathcal{L}) F_\ell. \tag{9}$$

Algorithm 1 Weight representation learning

Input: Data/loss $(\mathcal{D}, \mathcal{L})$; initial model $(\{W_\ell^{\text{init}}\}, \psi)$; subspace dimensions $\{d_\ell\}$; fixed seed.
Output: Exported weights and one global recovery threshold; retain the seed and dimensions.

1. Construct F_ℓ by Equation 10 ▷ seeded PRNG
2. $Z_\ell \leftarrow \sqrt{D_\ell/d_\ell}\, W_\ell^{\text{init}} F_\ell$
3. **for** training minibatch $\mathcal{B}$:
4. $W_\ell \leftarrow Z_\ell F_\ell^\top$ ▷ expand
5. $\mathcal{L}_\mathcal{B} \leftarrow \mathcal{L}(\{W_\ell\}, \psi; \mathcal{B})$
6. Update $\{Z_\ell\}$ and ψ using their loss gradients
7. **end for**
8. $\bar{W}_\ell \leftarrow Z_\ell F_\ell^\top$ ▷ store weights
9. $\tau_{\text{global}} \leftarrow \text{Calibrate}(\{\bar{W}_\ell, F_\ell\}_{\ell=1}^L)$
10. **return** $\{\bar{W}_\ell\}, \tau_{\text{global}}, \psi$

Algorithm 2 Test-time weight recovery

Input: Stored row v from row a of layer ℓ; subspace dimension d_ℓ; fixed seed; global threshold τ_{global}.
Output: Repaired or unchanged row v. Numerical failure returns v unchanged.

1. Construct/reuse F_ℓ by Equation 10
2. $(y, E, y^0, r_0, J, r) \leftarrow$Equation 13; $\tau \leftarrow \tau_{\text{global}}$
3. **if** $E = \varnothing$ and $\|r_0\|_\infty \le \tau$: **return** v
4. **while** $|J| < D_\ell - d_\ell$ $(k = 0, 1, \ldots)$:
5. $(j, \tau_k) \leftarrow$Equation 14 ▷ localize
6. **if** $|r_j| \le \tau_k$: **break**
7. $r' \leftarrow$Equation 15 on $J \cup \{j\}$
8. Accept by Equation 16, or **break**
9. **end while**
10. **if** $\text{Valid}_\ell(J)$: reconstruct $\widehat{w}_J$ by Equation 17; $v_J \leftarrow \widehat{w}_J$
11. **return** v ▷ selected entries only

Ordinary gradient-based optimization applies without a separate constraint step. Exported floating-point weights retain the original tensor shapes and layer computations, enabling recovery beyond classification.

Efficient generation of seeded subspaces. To address Challenge 2, the fixed seed and layer index generate one set of orthogonal factors shared by all output rows. With zero-padding matrix $E_\ell = [\mathrm{I}_{d_\ell}; 0] \in \mathbb{R}^{D_\ell \times d_\ell}$,

$$\begin{aligned} F_\ell &= \Lambda_\ell R_{\ell,T_\ell} \Pi_{\ell,T_\ell} \cdots R_{\ell,1} \Pi_{\ell,1} E_\ell, \\ T_\ell &= \max\{4, \lceil \log_2 D_\ell \rceil + 2\}. \end{aligned} \tag{10}$$

In each of T_ℓ mixing rounds, indexed by m, the $D_\ell \times D_\ell$ matrices $\Pi_{\ell,m}$ and $R_{\ell,m}$ permute entries and rotate disjoint pairs through pseudorandom angles uniform on $[0, 2\pi)$; an unpaired entry is unchanged. The diagonal matrix Λ_ℓ applies pseudorandom signs. Orthogonality gives $F_\ell^\top F_\ell = \mathrm{I}_{d_\ell}$ in exact arithmetic without QR. Factor initialization uses $O(T_\ell)$ tensor-operation calls per layer. Shared random values are cached across rows and repairs, with regeneration under identical PRNG and arithmetic conventions only when unavailable. Appendix C.3 accounts for tensor sizes and additional setup.

Calibration for floating-point recovery. To address Challenge 3, we calibrate one threshold shared by all selected layers and rows. For each clean stored row $\bar{w}_{\ell,a} = (\bar{W}_\ell)_{a,:}^\top$, first compute

$$\tau_{\ell,a}^{\text{clean}} = \|N_\ell \bar{w}_{\ell,a}\|_\infty + \epsilon_{\text{cal}}, \tag{11}$$

In this work, we use $\epsilon_{\text{cal}} = 8\epsilon_{\text{fp}}$ as a numerical safety margin, where $\epsilon_{\text{fp}} = \epsilon_{\text{FP32}} = 2^{-23}$ is the machine epsilon of the calibration and recovery arithmetic. With $\bar{W}$ denoting the vector of all clean selected weights, set

$$\tau_{\text{global}} = \max\left\{1.25 \max_{\ell,a} \tau_{\ell,a}^{\text{clean}},\ 64\epsilon_{\text{FP32}} \max(1, \|\bar{W}\|_\infty)\right\}. \tag{12}$$

The two terms cover the largest clean row residual and a roundoff margin scaled to the largest clean weight magnitude. Calibration uses the same frame operations as recovery, in arithmetic that exactly represents finite stored values. It is deterministic given the clean weights and numerical conventions, and runs at export or verified-clean loading without training data. Only a single value τ_{global} is retained and protected for calibration.

Iterative recovery of stored weights. To address Challenge 1 and Challenge 3, recovery first localizes suspect weights and then reconstructs only those entries. For row a of layer ℓ, let v contain the stored values and $z_{\ell,a} = (Z_\ell)_{a,:}^\top$. Finite observations satisfy $y = F_\ell z_{\ell,a} + \eta + e$, where η is clean numerical error and e includes faults and prior repair errors. Nonfinite entries are treated as known erasures. With Zero_J setting positions J to zero internally, initialize

$$\begin{aligned} y &= v, \qquad E = \{i : \neg\,\text{isfinite}(y_i)\}, \qquad y^0 = \text{Zero}_E(y), \\ r_0 &= N_\ell y^0, \qquad J = E, \qquad r = r_0. \end{aligned} \tag{13}$$

Every row uses the fixed threshold $\tau = \tau_{\text{global}}$ from Equation 12. If $E = \varnothing$ and $\|r_0\|_\infty \le \tau$, the row remains unchanged. Otherwise, normalized matching pursuit (Tropp & Gilbert, 2007) selects additional positions while $|J| < D_\ell - d_\ell$. At iteration k, with residual $r = r_k$, use

$$j = \arg\max_{i \notin J} \frac{|r_i|}{\sqrt{\max\{(N_\ell)_{ii}, \epsilon_{\text{fp}}\}}}, \qquad \tau_k = \max\{\tau, \min(8\,\text{median}_i\,|r_i|, 1.5\tau)\}. \tag{14}$$

The diagonal $(N_\ell)_{ii} = 1 - (P_\ell)_{ii}$ normalizes the score by frame geometry. We use eight times the median residual magnitude as the adaptive threshold, restricted to $[\tau, 1.5\tau]$ to keep it close to the global calibrated threshold. This temporary threshold requires no additional retained calibration value. Stop if $|r_j| \le \tau_k$. Otherwise, refit the tentative support $S = J \cup \{j\}$ against the original residual:

$$M_{\ell,S} = \mathrm{I}_{|S|} - F_{\ell,S} F_{\ell,S}^\top, \qquad h = M_{\ell,S}^\dagger (r_0)_S, \qquad r' = r_0 - N_\ell\,\text{scatter}_S(h). \tag{15}$$

Set subscripts select rows or entries, † denotes the pseudoinverse evaluated by numerical least squares, and $\text{scatter}_S(h)$ places h at S with zeros elsewhere. Accept the addition only when it sufficiently reduces the residual:

$$(J, r) \leftarrow \begin{cases} (S, r'), & \|r'\|_2 \le (1-\delta)\|r\|_2, \\ (J, r), & \text{otherwise (stop)}, \end{cases} \tag{16}$$

In this work, the required relative improvement is $\delta = \max\{10^{-4}, 64\epsilon_{\text{fp}}\}$, with $0 < \delta < 1$ for the recovery arithmetic. This rejects negligible reductions at the numerical precision used. For the final support J, reconstruction requires $0 < |J| \le D_\ell - d_\ell$ and a finite spectral condition number $\text{cond}_2(M_{\ell,J}) \le \kappa_{\max}$. We use $\kappa_{\max} = 10^7$ to reject ill-conditioned systems. Solve

$$g = F_\ell^\top\,\text{Zero}_J(y^0), \qquad M_{\ell,J}\xi = F_{\ell,J} g, \qquad \widehat{w}_J = F_{\ell,J}(g + F_{\ell,J}^\top \xi), \tag{17}$$

without forming an inverse, and write only $v_J \leftarrow \widehat{w}_J$ in the stored floating-point precision. Unselected bits remain unchanged; failed checks, failed solves, or nonfinite computations or stored outputs leave the entire row unchanged. For retained positions $U = \{1, \ldots, D_\ell\} \setminus J$, Appendix C.4 establishes equivalence to retained-row least squares, and Appendix C.5 analyzes the cost. Restoration is conditional on successful localization (Appendix C.6), and subsequent faults act on the actual repaired state (Equation 2).

Complete training and recovery procedures are presented in Algorithm 1 and Algorithm 2. Training steps apply to every selected layer, with minibatch loss $\mathcal{L}_\mathcal{B}$. Calibrate returns the single scalar τ_{global} using Equation 11 and Equation 12; $\text{Valid}_\ell(J)$ checks support size and conditioning. Recovery commits only finite repairs.

4 Experiments

4.1 Experimental Setup

We evaluate predictive performance, robustness, and training and recovery costs across datasets, architectures, and tasks. All experiments run on an NVIDIA H200 GPU. Test-time efficiency measurements, including recovery runtime, restrict CUDA execution to 20 streaming multiprocessors (SMs) using a CUDA Green Context, with GPU power capped at 200 W. This configuration targets a nominal FP32 compute capacity and power budget comparable to the RTX 2080 used in prior edge-inference studies (Verma et al., 2021), providing a controlled evaluation under constrained computational resources.

Datasets and model architectures. The classification study combines CIFAR-10 and CIFAR-100 (Krizhevsky et al., 2009), SVHN (Netzer et al., 2011), and GTSRB (Stallkamp et al., 2011) with ResNet-18 and ResNet-34 (He et al., 2016), DenseNet-121 (Huang et al., 2017), VGG-11 (Simonyan & Zisserman, 2015), and ViT-B/16 (Dosovitskiy et al., 2021), covering 20 dataset–architecture settings. We additionally evaluate ECG5000 (Dau et al., 2019) time-series classification with an LSTM (Hochreiter & Schmidhuber, 1997), CIFAR-10 image reconstruction with a U-Net (Ronneberger et al., 2015), WikiText-2 (Merity et al., 2017) language modeling with an LSTM, and VOC 2007 (Everingham et al., 2010) object detection with YOLOv8n (Jocher et al., 2023).

Table 1: **Robust accuracy after one random injection.** FP32 accuracy (%) is reported as mean$_{\pm \mathrm{SD}}$ over three independent fault realizations, after recovery where applicable. Column headers give the BER applied independently to eligible bits. Bold marks the highest displayed mean within each dataset–architecture–BER comparison, including ties.

Dataset	Method	Architectures														
		ResNet-18			ResNet-34			DenseNet-121			VGG-11			ViT-B/16		
		10^{-6}	10^{-5}	10^{-4}	10^{-6}	10^{-5}	10^{-4}	10^{-6}	10^{-5}	10^{-4}	10^{-6}	10^{-5}	10^{-4}	10^{-6}	10^{-5}	10^{-4}
CIFAR-10	None	10.0±0.0	10.0±0.0	10.0±0.0	10.0±0.0	10.0±0.0	10.0±0.0	9.7±0.5	10.0±0.0	10.0±0.0	10.3±1.0	10.0±0.0	10.0±0.0	10.0±0.0	10.0±0.0	10.0±0.0
	OCM-Ozdenizci	10.7±2.0	10.0±0.0	10.0±0.0	10.0±0.0	10.0±0.0	10.0±0.0	9.0±1.2	10.0±0.0	10.0±0.0	10.0±0.0	10.0±0.0	10.0±0.0	10.0±0.0	10.0±0.0	10.0±0.0
	OCM-Hadamard	9.4±1.0	10.0±0.0	10.0±0.0	10.0±0.0	10.0±0.0	10.0±0.0	9.4±0.6	10.0±0.0	10.0±0.0	10.0±0.0	10.0±0.0	10.0±0.0	10.0±0.0	10.0±0.0	10.0±0.0
	Real-ECC	36.8±46.7	10.0±0.0	10.0±0.0	**91.1±0.0**	10.0±0.0	10.0±0.0	**89.8±0.0**	10.0±0.0	10.0±0.0	**86.2±0.0**	10.0±0.0	10.0±0.0	10.0±0.0	10.0±0.0	10.0±0.0
	LoDiSS (Ours) w/o recovery	10.0±0.0	10.0±0.0	10.0±0.0	10.0±0.0	10.0±0.0	10.0±0.0	9.9±0.2	10.0±0.0	10.0±0.0	10.6±0.9	10.0±0.0	10.0±0.0	10.0±0.0	10.0±0.0	10.0±0.0
	LoDiSS (Ours)	**89.6±0.0**	**89.6±0.0**	**89.5±0.2**	90.1±0.0	**90.1±0.0**	**37.4±45.5**	88.5±0.0	**88.5±0.0**	**88.3±0.3**	85.0±0.0	**85.0±0.0**	**84.7±0.1**	**66.0±0.0**	**66.0±0.0**	**47.2±32.2**
CIFAR-100	None	1.0±0.0	1.0±0.0	1.0±0.0	1.0±0.0	1.0±0.0	1.0±0.0	0.9±0.2	1.0±0.0	1.0±0.0	1.1±0.1	1.0±0.0	1.0±0.0	1.0±0.0	1.0±0.0	1.0±0.0
	OCM-Ozdenizci	1.0±0.1	1.0±0.0	1.0±0.0	1.0±0.0	1.0±0.0	1.0±0.0	0.9±0.1	1.0±0.0	1.0±0.0	1.0±0.0	1.0±0.0	1.0±0.0	1.0±0.0	1.0±0.0	1.0±0.0
	OCM-Hadamard	1.1±0.1	1.0±0.0	1.0±0.0	1.0±0.0	1.0±0.0	1.0±0.0	1.0±0.0	1.0±0.0	1.0±0.0	1.8±0.8	1.0±0.0	1.0±0.0	1.0±0.0	1.0±0.0	1.0±0.0
	Real-ECC	43.9±37.0	1.0±0.0	1.0±0.0	44.5±37.3	1.0±0.0	1.0±0.0	**62.4±0.0**	1.3±0.5	1.0±0.0	44.7±0.0	1.0±0.0	1.0±0.0	1.0±0.0	1.0±0.0	1.0±0.0
	LoDiSS (Ours) w/o recovery	1.0±0.0	1.0±0.0	1.0±0.0	1.0±0.0	1.0±0.0	1.0±0.0	1.2±0.2	1.0±0.0	1.0±0.0	2.6±2.5	1.0±0.0	1.0±0.0	1.0±0.0	1.0±0.0	1.0±0.0
	LoDiSS (Ours)	**63.4±0.0**	**63.4±0.0**	**63.2±0.3**	**64.0±0.0**	**64.0±0.1**	**23.9±34.3**	60.5±0.0	**60.5±0.0**	**59.6±0.3**	**46.4±0.0**	**46.4±0.0**	**46.2±0.3**	**34.4±0.0**	**34.4±0.0**	**23.3±19.2**
SVHN	None	6.7±0.0	6.7±0.0	6.7±0.0	6.7±0.0	6.7±0.0	6.7±0.0	8.4±2.9	6.7±0.0	6.7±0.0	16.1±10.1	6.7±0.0	6.7±0.0	9.2±0.0	6.7±0.0	6.7±0.0
	OCM-Ozdenizci	9.8±5.4	6.7±0.0	6.7±0.0	6.7±0.0	6.7±0.0	6.7±0.0	6.8±0.1	6.7±0.0	6.7±0.0	11.0±7.4	6.7±0.0	6.7±0.0	19.6±0.0	6.7±0.0	6.7±0.0
	OCM-Hadamard	8.6±3.3	6.7±0.0	6.7±0.0	6.7±0.0	6.7±0.0	6.7±0.0	8.9±4.1	6.7±0.0	6.7±0.0	11.0±7.4	6.7±0.0	6.7±0.0	19.6±0.0	6.7±0.0	6.7±0.0
	Real-ECC	42.0±47.2	6.7±0.0	6.7±0.0	70.7±44.3	6.7±0.0	6.7±0.0	**95.9±0.0**	6.7±0.0	6.7±0.0	**95.3±0.0**	6.7±0.0	6.7±0.0	9.2±0.0	9.2±0.0	6.7±0.0
	LoDiSS (Ours) w/o recovery	6.2±0.8	6.7±0.0	6.7±0.0	6.7±0.0	6.7±0.0	6.7±0.0	7.7±1.8	6.7±0.0	6.7±0.0	12.9±3.8	6.7±0.0	6.7±0.0	9.7±0.0	6.7±0.0	6.7±0.0
	LoDiSS (Ours)	**95.6±0.0**	**95.6±0.0**	**95.5±0.0**	**96.1±0.0**	**96.1±0.0**	**44.6±45.0**	95.5±0.0	**95.5±0.0**	**95.5±0.0**	94.4±0.0	**94.4±0.0**	**94.3±0.0**	**89.9±0.0**	**89.9±0.0**	**63.2±46.3**
GTSRB	None	2.1±2.8	0.5±0.0	0.5±0.0	0.5±0.0	0.5±0.0	0.5±0.0	0.5±0.0	0.5±0.0	0.5±0.0	22.6±31.3	0.5±0.0	0.5±0.0	3.6±0.0	0.5±0.0	0.5±0.0
	OCM-Ozdenizci	2.0±2.6	0.5±0.0	0.5±0.0	0.5±0.0	0.5±0.0	0.5±0.0	0.5±0.0	0.5±0.0	0.5±0.0	2.3±3.2	0.5±0.0	0.5±0.0	5.0±0.0	0.5±0.0	0.5±0.0
	OCM-Hadamard	2.7±3.8	0.5±0.0	0.5±0.0	0.5±0.0	0.5±0.0	0.5±0.0	0.8±0.3	0.5±0.0	0.5±0.0	2.1±2.9	0.5±0.0	0.5±0.0	3.6±0.0	0.5±0.0	0.5±0.0
	Real-ECC	66.1±55.6	0.5±0.0	0.5±0.0	66.5±55.1	0.5±0.0	0.5±0.0	**97.9±0.0**	0.5±0.0	0.5±0.0	93.3±0.0	0.5±0.0	0.5±0.0	5.0±0.0	5.0±0.0	0.5±0.0
	LoDiSS (Ours) w/o recovery	2.0±2.6	0.5±0.0	0.5±0.0	0.5±0.0	0.5±0.0	0.5±0.0	6.7±8.4	0.5±0.0	0.5±0.0	2.6±2.1	0.5±0.0	0.5±0.0	5.0±0.0	0.5±0.0	0.5±0.0
	LoDiSS (Ours)	**97.7±0.0**	**97.7±0.0**	**97.6±0.1**	**97.4±0.0**	**97.4±0.0**	**41.5±48.8**	96.1±0.0	**96.1±0.0**	**96.0±0.1**	**93.5±0.0**	**93.5±0.0**	**93.4±0.0**	**75.1±0.0**	**75.1±0.0**	**51.6±40.4**

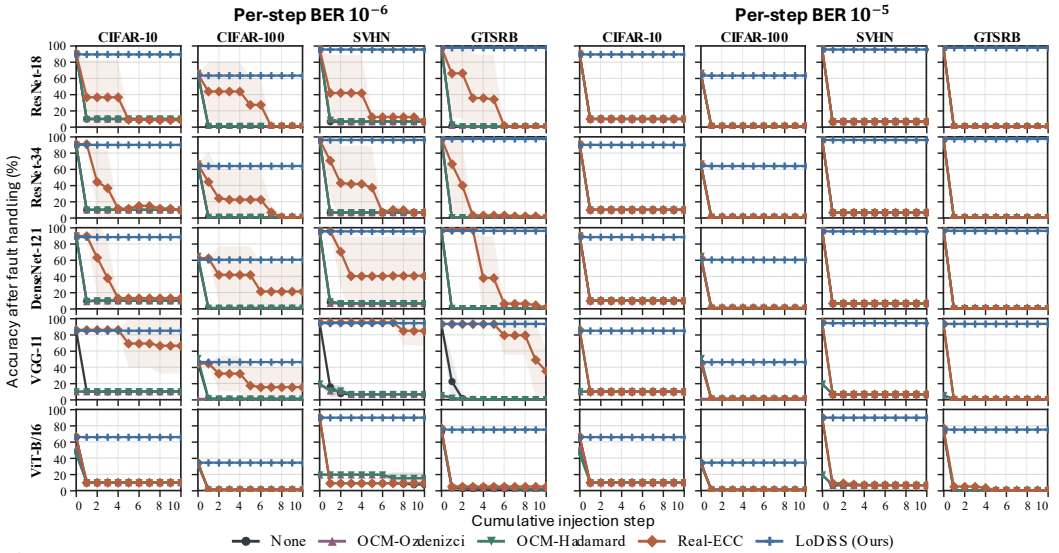


Figure 2: **Robust accuracy over ten random injection–recovery cycles.** Rows show architectures and columns show datasets, with BER 10^{-6} on the left and 10^{-5} on the right. Step 0 gives clean accuracy. Lines and shaded bands show mean FP32 accuracy and standard deviation over three fault trajectories. Each repair is retained before the next injection.

Defense baselines. We compare LoDiSS with None, an undefended model, OCM-Ozdenizci (Özdenizci & Legenstein, 2022) and OCM-Hadamard (Yu et al., 2024), and Real-ECC (Li et al., 2026). Real-ECC uses its Type-1 weight-memory formulation with rowwise groups and batched noise-tolerant ℓ_1 recovery, matched to LoDiSS in deployed weight count and effective degrees of freedom. OCM code lengths are 64 for CIFAR-10/SVHN and 512 for CIFAR-100/GTSRB. The additional tasks compare None, Real-ECC, and LoDiSS, whose recovery does not require class-code outputs. A LoDiSS ablation disables recovery while preserving the learned weights and fault sequence.

Evaluated threats and metrics. Following Section 2, single random injections use BER $p \in \{10^{-6}, 10^{-5}, 10^{-4}\}$, overlapping the lower measured SRAM error-rate range in Stutz et al. (2021). Ten-cycle random evaluations use the two lower rates, selected after examining the broader sweep and applied uniformly across methods and tasks. Adversarial evaluations use the bit-flip attack (BFA) and adaptive BFA, which select faults using pre-recovery and post-recovery loss, respectively. We evaluate 16 cumulative flips with recovery applied to evaluation copies, and ten repair cycles with 1, 2, 4, or 8 flips per cycle, retaining the repaired state between cycles. Clean accuracy is measured before corruption, while robust accuracy follows injection and recovery where available. Clean-accuracy retention is the ratio of robust to clean accuracy. ECG classification, image reconstruction, language modeling, and object detection use macro-F1, peak signal-to-noise ratio (PSNR), perplexity, and

Table 2: **Robust accuracy after 16 cumulative adversarial bit flips.** FP32 accuracy (%) is reported as mean$_{\pm \mathrm{SD}}$ over three paired attack-calibration splits. Flips accumulate without committing repairs; recovery is applied to evaluation copies. BFA and Adapt. denote results under regular and adaptive adversarial bit-flip attacks, respectively. Dashes indicate unavailable settings. Bold marks the highest displayed mean within each dataset–architecture–attack comparison.

Dataset	Method	ResNet-18		ResNet-34		DenseNet-121		VGG-11		ViT-B/16	
		BFA	Adapt.	BFA	Adapt.	BFA	Adapt.	BFA	Adapt.	BFA	Adapt.
CIFAR-10	None	$0.0_{\pm 0.0}$	$0.0_{\pm 0.0}$	$0.0_{\pm 0.0}$	$0.0_{\pm 0.0}$	$0.0_{\pm 0.0}$	$0.0_{\pm 0.0}$	$0.0_{\pm 0.0}$	$0.0_{\pm 0.0}$	$6.6_{\pm 5.7}$	$6.6_{\pm 5.7}$
	OCM-Ozdenizci	$0.1_{\pm 0.1}$	—	$0.1_{\pm 0.1}$	—	$0.1_{\pm 0.0}$	—	$2.3_{\pm 2.0}$	—	$26.9_{\pm 11.7}$	—
	OCM-Hadamard	$0.0_{\pm 0.0}$	—	$0.5_{\pm 0.4}$	—	$0.1_{\pm 0.0}$	—	$0.6_{\pm 1.0}$	—	$14.8_{\pm 12.2}$	—
	Real-ECC	$30.5_{\pm 52.2}$	$0.0_{\pm 0.0}$	$60.7_{\pm 52.5}$	$0.0_{\pm 0.0}$	$0.2_{\pm 0.4}$	$0.2_{\pm 0.4}$	$31.7_{\pm 47.4}$	$0.0_{\pm 0.0}$	$0.0_{\pm 0.0}$	$0.0_{\pm 0.0}$
	LoDiSS (Ours)	$\mathbf{89.6_{\pm 0.0}}$	$\mathbf{89.6_{\pm 0.0}}$	$\mathbf{90.5_{\pm 0.0}}$	$\mathbf{90.5_{\pm 0.0}}$	$\mathbf{88.6_{\pm 0.0}}$	$\mathbf{59.1_{\pm 51.2}}$	$\mathbf{85.4_{\pm 0.0}}$	$\mathbf{85.4_{\pm 0.0}}$	$\mathbf{49.9_{\pm 29.4}}$	$\mathbf{49.9_{\pm 29.4}}$
CIFAR-100	None	$0.0_{\pm 0.0}$	$0.0_{\pm 0.0}$	$0.0_{\pm 0.1}$	$0.0_{\pm 0.1}$	$0.0_{\pm 0.0}$	$0.0_{\pm 0.0}$	$0.0_{\pm 0.0}$	$0.0_{\pm 0.0}$	$0.6_{\pm 1.0}$	$0.6_{\pm 1.0}$
	OCM-Ozdenizci	$0.0_{\pm 0.0}$	—	$0.7_{\pm 0.3}$	—	$0.7_{\pm 0.2}$	—	$0.1_{\pm 0.2}$	—	$26.0_{\pm 6.9}$	—
	OCM-Hadamard	$0.0_{\pm 0.0}$	—	$0.5_{\pm 0.2}$	—	$0.3_{\pm 0.3}$	—	$0.0_{\pm 0.0}$	—	$25.7_{\pm 10.9}$	—
	Real-ECC	$0.1_{\pm 0.2}$	$0.0_{\pm 0.0}$	$22.3_{\pm 37.9}$	$0.0_{\pm 0.0}$	$0.0_{\pm 0.0}$	$0.0_{\pm 0.0}$	$15.0_{\pm 25.7}$	$0.0_{\pm 0.0}$	$0.0_{\pm 0.0}$	$0.0_{\pm 0.0}$
	LoDiSS (Ours)	$\mathbf{64.2_{\pm 0.0}}$	$\mathbf{21.9_{\pm 36.7}}$	$\mathbf{64.4_{\pm 0.0}}$	$\mathbf{64.4_{\pm 0.0}}$	$\mathbf{60.7_{\pm 0.0}}$	$\mathbf{40.5_{\pm 35.0}}$	$\mathbf{47.2_{\pm 0.0}}$	$\mathbf{47.2_{\pm 0.0}}$	$\mathbf{35.5_{\pm 0.0}}$	$\mathbf{35.5_{\pm 0.0}}$
SVHN	None	$0.0_{\pm 0.1}$	$0.0_{\pm 0.1}$	$0.0_{\pm 0.0}$	$0.0_{\pm 0.0}$	$2.0_{\pm 3.3}$	$2.0_{\pm 3.3}$	$0.1_{\pm 0.1}$	$0.1_{\pm 0.1}$	$0.0_{\pm 0.0}$	$0.0_{\pm 0.0}$
	OCM-Ozdenizci	$0.0_{\pm 0.0}$	—	$17.0_{\pm 5.8}$	—	$10.5_{\pm 7.3}$	—	$5.4_{\pm 1.0}$	—	$7.3_{\pm 2.1}$	—
	OCM-Hadamard	$0.0_{\pm 0.0}$	—	$18.0_{\pm 5.0}$	—	$6.7_{\pm 0.5}$	—	$6.5_{\pm 0.7}$	—	$7.4_{\pm 2.0}$	—
	Real-ECC	$4.2_{\pm 6.5}$	$4.2_{\pm 6.5}$	$4.8_{\pm 8.3}$	$0.0_{\pm 0.0}$	$2.6_{\pm 4.5}$	$0.0_{\pm 0.0}$	$48.1_{\pm 40.7}$	$8.4_{\pm 14.6}$	$30.1_{\pm 52.1}$	$0.0_{\pm 0.0}$
	LoDiSS (Ours)	$\mathbf{95.9_{\pm 0.0}}$	$\mathbf{69.2_{\pm 46.2}}$	$\mathbf{96.1_{\pm 0.0}}$	$\mathbf{96.1_{\pm 0.0}}$	$\mathbf{95.9_{\pm 0.0}}$	$\mathbf{95.9_{\pm 0.0}}$	$\mathbf{67.7_{\pm 47.2}}$	$\mathbf{67.7_{\pm 47.2}}$	$\mathbf{90.8_{\pm 0.0}}$	$\mathbf{90.8_{\pm 0.0}}$
GTSRB	None	$0.0_{\pm 0.0}$	$0.0_{\pm 0.0}$	$0.0_{\pm 0.0}$	$0.0_{\pm 0.0}$	$0.0_{\pm 0.0}$	$0.0_{\pm 0.0}$	$0.0_{\pm 0.0}$	$0.0_{\pm 0.0}$	$0.9_{\pm 1.6}$	$0.9_{\pm 1.6}$
	OCM-Ozdenizci	$0.0_{\pm 0.0}$	—	$12.7_{\pm 2.3}$	—	$0.0_{\pm 0.0}$	—	$0.3_{\pm 0.2}$	—	$14.4_{\pm 20.3}$	—
	OCM-Hadamard	$0.0_{\pm 0.0}$	—	$9.5_{\pm 6.4}$	—	$0.0_{\pm 0.0}$	—	$0.7_{\pm 0.7}$	—	$39.9_{\pm 23.3}$	—
	Real-ECC	$0.7_{\pm 1.2}$	$0.0_{\pm 0.0}$	$5.7_{\pm 8.3}$	$0.0_{\pm 0.1}$	$0.5_{\pm 0.8}$	$0.0_{\pm 0.0}$	$0.0_{\pm 0.0}$	$0.0_{\pm 0.0}$	$0.0_{\pm 0.0}$	$0.0_{\pm 0.0}$
	LoDiSS (Ours)	$\mathbf{65.4_{\pm 56.6}}$	$\mathbf{65.4_{\pm 56.6}}$	$\mathbf{98.3_{\pm 0.0}}$	$\mathbf{37.2_{\pm 53.2}}$	$\mathbf{96.7_{\pm 0.0}}$	$\mathbf{69.1_{\pm 47.8}}$	$\mathbf{69.1_{\pm 43.1}}$	$\mathbf{40.2_{\pm 46.9}}$	$\mathbf{76.3_{\pm 0.0}}$	$\mathbf{76.3_{\pm 0.0}}$

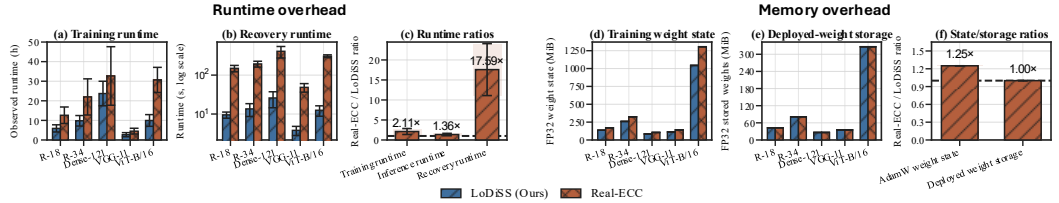


Figure 3: **Training and recovery costs.** (a,b) show measured training and recovery times; (c) shows paired Real-ECC/LoDiSS runtime ratios. Bars and whiskers indicate means and standard deviations, with results for each architecture summarizing four datasets. Training time includes clean evaluation; recovery time is averaged over 60 recovery runs per setting. (d–f) show FP32 weight-related training state, deployed weight storage, and their ratios.

mean average precision over intersection-over-union thresholds 0.50–0.95 (mAP50:95), respectively. Each setting uses one training run and three fault realizations or attack-calibration splits, so standard deviations reflect fault or attack variability.

Implementation details. Unless noted otherwise, weights use FP32 and $\rho = d/D = 0.8$, with representation dimensions matched between LoDiSS and Real-ECC. Classification models train for 150 epochs with AdamW (Loshchilov & Hutter, 2019), batch size 128, learning rate 10^{-4}, and weight decay 5×10^{-4}. Additional tasks train for 200 epochs with the same optimizer and weight decay at learning rate 10^{-3}. YOLOv8n uses pretrained initialization; the other additional-task models train from scratch. For adversarial bit-flip attacks, we use 64 labeled examples excluded from evaluation to search all bits of the 100 weights per layer with the largest absolute loss gradients.

4.2 Experimental Results

Does LoDiSS preserve accuracy under random bit flips in floating-point weights? LoDiSS achieves the highest robust accuracy in all 20 dataset–architecture settings at BERs 10^{-5} and 10^{-4} (Table 1). Averaged equally across these settings, its gains over Real-ECC are approximately 76 and 62 pp, respectively. Disabling recovery on the same learned weights causes severe degradation, showing that active repair is essential to this improvement. At 10^{-4}, ResNet-34 and ViT-B/16 show larger accuracy losses and variability, while the other 12 settings remain within 0.9 pp of their results at 10^{-5}. The comparison demonstrates effective floating-point weight recovery across architectures and datasets, addressing Challenge 3 while showing how performance changes under heavier corruption.

Does LoDiSS remain robust as random bit flips accumulate? LoDiSS retains approximately 100% of its initial accuracy across all 20 settings after ten recovery cycles at BERs 10^{-6} and 10^{-5} (Figure 2). Its accuracy remains nearly constant as new faults occur, while Real-ECC deteriorates over successive cycles and the OCM-based methods provide little protection in these floating-point settings. Each cycle starts from the weights left by the previous repair, so this result includes the effects of

Table 3: **Task performance after five and ten random injection–recovery cycles.** Column headers give the per-injection BER. Clean values precede corruption; subsequent values are mean$_{\pm \mathrm{SD}}$ over three fault trajectories, after recovery where applicable. fail $(3/3)$ denotes nonfinite outputs in all three trajectories. Arrows indicate the preferred direction, and bold marks the best displayed value within each task and column. OCM-based methods are omitted because class-code outputs do not directly support evaluated non-classification tasks.

Task	Metric	Method	Clean	After 5 injections		After 10 injections	
				10^{-6}	10^{-5}	10^{-6}	10^{-5}
ECG5000	Macro-F1 ↑	None	0.595	$0.595_{\pm 0.000}$	$0.561_{\pm 0.033}$	$0.594_{\pm 0.001}$	$0.502_{\pm 0.065}$
		Real-ECC	0.582	$0.582_{\pm 0.000}$	$0.561_{\pm 0.014}$	$0.582_{\pm 0.000}$	$0.469_{\pm 0.151}$
		LoDiSS (Ours)	**0.631**	$\mathbf{0.631_{\pm 0.000}}$	$\mathbf{0.631_{\pm 0.000}}$	$\mathbf{0.631_{\pm 0.000}}$	$\mathbf{0.631_{\pm 0.000}}$
CIFAR-10 reconstruction	PSNR (dB) ↑	None	**50.88**	fail (3/3)	fail (3/3)	fail (3/3)	fail (3/3)
		Real-ECC	49.16	$49.16_{\pm 0.00}$	fail (3/3)	$19.74_{\pm 25.48}$	fail (3/3)
		LoDiSS (Ours)	50.59	$\mathbf{50.59_{\pm 0.00}}$	$\mathbf{50.59_{\pm 0.00}}$	$\mathbf{50.59_{\pm 0.00}}$	$\mathbf{50.59_{\pm 0.00}}$
WikiText-2	Perplexity ↓	None	211.8	fail (3/3)	fail (3/3)	fail (3/3)	fail (3/3)
		Real-ECC	161.8	fail (3/3)	fail (3/3)	fail (3/3)	fail (3/3)
		LoDiSS (Ours)	**154.0**	$\mathbf{154.0_{\pm 0.0}}$	$\mathbf{154.0_{\pm 0.0}}$	$\mathbf{154.0_{\pm 0.0}}$	$\mathbf{154.0_{\pm 0.0}}$
VOC 2007	mAP50:95 ↑	None	**0.170**	$0.000_{\pm 0.000}$	$0.000_{\pm 0.000}$	$0.000_{\pm 0.000}$	$0.000_{\pm 0.000}$
		Real-ECC	0.168	$0.069_{\pm 0.088}$	$0.000_{\pm 0.000}$	$0.013_{\pm 0.023}$	$0.000_{\pm 0.000}$
		LoDiSS (Ours)	0.168	$\mathbf{0.168_{\pm 0.000}}$	$\mathbf{0.168_{\pm 0.000}}$	$\mathbf{0.168_{\pm 0.000}}$	$\mathbf{0.168_{\pm 0.000}}$

any reconstruction error. Maintaining performance throughout these trajectories demonstrates that repeated recovery can prevent persistent faults from progressively degrading inference over the evaluated cycles.

Does LoDiSS remain robust under long-term adversarial bit-flip attacks? We evaluate BFA and adaptive BFA with 16 cumulative weight-bit flips before recovery, without physical injection delays or restrictions on flippable locations. This grants greater attacker control than DeepHammer's hardware-constrained attacks, which require roughly a day of offline memory profiling followed by 66–96 seconds online to induce 13–20 selected VGG-16 flips (Yao et al., 2020), while LoDiSS averages approximately 4 seconds for VGG-11 recovery (Figure 3). LoDiSS achieves the highest mean robust accuracy in every setting under both BFA and adaptive BFA, improving over Real-ECC by approximately 64 pp on average for each attack (Table 2). These fixed-budget results evaluate recovery on copies of the corrupted model. We further test an adversary that repeatedly injects 1, 2, 4, or 8 selected bit flips before each repair. LoDiSS maintains higher accuracy than None and Real-ECC under these repeated attacks, as shown in the Appendix (Figure 5).

Does LoDiSS improve training and test-time recovery efficiency? With matched representation dimensions and deployed weight counts, LoDiSS provides observed mean paired speedups of $2.11\times$ for training and $17.59\times$ for recovery over Real-ECC (Figure 3). Direct subspace learning also reduces storage for trainable weights, gradients, and optimizer moments by approximately 20%, while deployed weight storage remains unchanged. Together, the results show lower training-state requirements and substantially faster repair, addressing Challenge 1 and reducing the computational cost of repeated recovery.

Does LoDiSS support recovery across different prediction tasks? LoDiSS preserves its original task performance to the displayed precision in all four tasks after both five and ten random injections at either BER (Table 3). After ten injections at 10^{-5}, Real-ECC and None lose approximately 19.4% and 15.6% of their initial ECG macro-F1, respectively, while LoDiSS shows no displayed loss. Both baselines produce nonfinite outputs in all three image-reconstruction and language-modeling trajectories and reach zero detection mAP under this condition. LoDiSS maintains the reconstruction, language-modeling, and detection metrics throughout. These results demonstrate that recovery operates across different prediction tasks without relying on classification-specific prediction tasks.

5 Conclusions

LoDiSS learns recoverable floating-point weights in low-dimensional seeded subspaces, combining compact training representations with recovery constraints that can be efficiently regenerated at test time. Calibrated iterative recovery supports reliable and fast repair of damaged weights without training data or a clean model copy. Across the evaluated architectures and tasks, LoDiSS sustains performance under repeated random faults and improves robustness to adversarial and recovery-aware attacks, while reducing weight-related training state and observed training and recovery times relative to the existing baseline. These results support reliable long-term deployment of floating-point models on edge devices with unreliability, broadening practical deployment choices.

AI USE STATEMENT

In preparing this manuscript, we used generative AI tools to assist with editing author-written text, including grammar correction, typo detection, and improvements to clarity and readability. The authors reviewed and revised the suggested edits to preserve the intended meaning and consistency with the technical content. The authors take responsibility for the final manuscript and its claims.

ETHICS STATEMENT

This work aims to improve the reliability of machine learning systems. We emphasize responsible use of datasets and models, with consideration of data privacy, applicable licenses, and potential societal impacts. The fault evaluations are intended to support the development of more reliable systems.

REPRODUCIBILITY STATEMENT

The main paper describes the proposed method, threat models, evaluation protocols, and implementation settings. The appendices provide supporting theoretical analysis and additional experimental details. Together, these materials document the procedures used to obtain the reported results and support their reproduction.

REFERENCES

Harshavardhan Adepu, Zhanpeng Zeng, Li Zhang, and Vikas Singh. FrameQuant: Flexible low-bit quantization for transformers. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 203–227. PMLR, 21–27 Jul 2024.

Anish Athalye, Nicholas Carlini, and David Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *International conference on machine learning*, pp. 274–283. PMLR, 2018.

Han Cai, Junyan Li, Muyan Hu, Chuang Gan, and Song Han. Efficientvit: Lightweight multi-scale attention for high-resolution dense prediction. In *2023 IEEE/CVF international conference on computer vision (ICCV)*, pp. 17256–17267. IEEE, 2023.

Emmanuel J Candes and Paige A Randall. Highly robust error correction by convex programming. *IEEE Transactions on Information Theory*, 54(7):2829–2840, 2008. doi: 10.1109/TIT.2008.924688.

Emmanuel J Candes and Terence Tao. Decoding by linear programming. *IEEE transactions on information theory*, 51(12):4203–4215, 2005.

Peter G Casazza, Andreas Heinecke, Felix Krahmer, and Gitta Kutyniok. Optimally sparse frames. *IEEE transactions on information Theory*, 57(11):7279–7287, 2011.

Kevin K Chang, A Giray Yağlıkçı, Saugata Ghose, Aditya Agrawal, Niladrish Chatterjee, Abhijith Kashyap, Donghyuk Lee, Mike O'Connor, Hasan Hassan, and Onur Mutlu. Understanding reduced-voltage operation in modern dram devices: Experimental characterization, analysis, and mechanisms. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 1(1): 1–42, 2017.

Kamran Chitsaz, Goncalo Mordido, Jean-Pierre David, and François Leduc-Primeau. Training dnns resilient to adversarial and random bit-flips by learning quantization ranges. *Transactions on Machine Learning Research*, 2023.

Hoang Anh Dau, Anthony Bagnall, Kaveh Kamgar, Chin-Chia Michael Yeh, Yan Zhu, Shaghayegh Gharghabi, Chotirat Ann Ratanamahatana, and Eamonn Keogh. The ucr time series archive. *IEEE/CAA Journal of Automatica Sinica*, 6(6):1293–1305, 2019.

Misha Denil, Babak Shakibi, Laurent Dinh, Marc'Aurelio Ranzato, and Nando De Freitas. Predicting parameters in deep learning. *Advances in neural information processing systems*, 26, 2013.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021.

Mathieu Dumont, Kevin Hector, Pierre-Alain Moellic, Jean-Max Dutertre, and Simon Pontié. Evaluation of parameter-based attacks against embedded neural networks with laser injection. In *International Conference on Computer Safety, Reliability, and Security*, pp. 259–272. Springer, 2023.

Steven K. Esser, Jeffrey L. McKinstry, Deepika Bablani, Rathinakumar Appuswamy, and Dharmendra S. Modha. Learned step size quantization. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=rkgO66VKDS.

Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2): 303–338, 2010.

Matthew Fickus and Dustin G Mixon. Numerically erasure-robust frames. *Linear Algebra and its Applications*, 437(6):1394–1407, 2012.

Anteneh Gebregiorgis, Saman Kiamehr, Fabian Oboril, Rajendra Bishnoi, and Mehdi B Tahoori. A cross-layer analysis of soft error, aging and process variation in near threshold computing. In *2016 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 205–210. IEEE, 2016.

Hui Guan, Lin Ning, Zhen Lin, Xipeng Shen, Huiyang Zhou, and Seung-Hwan Lim. In-place zero-space memory protection for cnn. *Advances in Neural Information Processing Systems*, 32, 2019.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pp. 1026–1034, 2015.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8): 1735–1780, 1997.

Sanghyun Hong, Pietro Frigo, Yiğitcan Kaya, Cristiano Giuffrida, and Tudor Dumitraș. Terminal brain damage: Exposing the graceless degradation in deep neural networks under hardware fault attacks. In *28th USENIX Security Symposium (USENIX Security 19)*, pp. 497–514, 2019.

Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *2017 IEEE conference on computer vision and pattern recognition (CVPR)*, pp. 2261–2269. Ieee, 2017.

Glenn Jocher, Ayush Chaurasia, and Jing Qiu. Ultralytics yolov8, 2023. URL https://github.com/ultralytics/ultralytics.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Andrey Kuzmin, Mart Van Baalen, Yuwei Ren, Markus Nagel, Jorn Peters, and Tijmen Blankevoort. Fp8 quantization: The power of the exponent. *Advances in Neural Information Processing Systems*, 35:14651–14662, 2022.

Chunyuan Li, Heerad Farkhoor, Rosanne Liu, and Jason Yosinski. Measuring the intrinsic dimension of objective landscapes. In *International Conference on Learning Representations*, 2018.

Ziqing Li, Myung Cho, Qiutong Jin, and Weiyu Xu. Repair brain damage: Real-numbered error correction code for neural network. *arXiv preprint arXiv:2602.00076*, 2026.

Jamie Liu, Ben Jaiyen, Yoongu Kim, Chris Wilkerson, and Onur Mutlu. An experimental study of data retention behavior in modern dram devices: Implications for retention time profiling mechanisms. *ACM SIGARCH Computer Architecture News*, 41(3):60–71, 2013.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. In *International Conference on Learning Representations*, 2017.

Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Baolin Wu, Andrew Y Ng, et al. Reading digits in natural images with unsupervised feature learning. In *NIPS workshop on deep learning and unsupervised feature learning*, volume 2011, pp. 4. Granada, 2011.

Ozan Özdenizci and Robert Legenstein. Improving robustness against stealthy weight bit-flip attacks by output code matching. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 13378–13387. IEEE, 2022.

Rachmad Vidya Wicaksana Putra, Muhammad Abdullah Hanif, and Muhammad Shafique. Softsnn: Low-cost fault tolerance for spiking neural network accelerators under soft errors. In *Proceedings of the 59th ACM/IEEE Design Automation Conference*, pp. 151–156, 2022.

Adnan Siraj Rakin, Zhezhi He, and Deliang Fan. Bit-flip attack: Crushing neural network with progressive bit search. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 1211–1220. IEEE, 2019.

Paolo Rech. Artificial neural networks for space and safety-critical applications: Reliability issues and potential solutions. *IEEE Transactions on Nuclear Science*, 71(4):377–404, 2024.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pp. 234–241. Springer, 2015.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015.

Johannes Stallkamp, Marc Schlipsing, Jan Salmen, and Christian Igel. The german traffic sign recognition benchmark: a multi-class classification competition. In *The 2011 international joint conference on neural networks*, pp. 1453–1460. IEEE, 2011.

David Stutz, Nandhini Chandramoorthy, Matthias Hein, and Bernt Schiele. Bit error robustness for energy-efficient dnn accelerators. *Proceedings of Machine Learning and Systems*, 3:569–598, 2021.

Joel A Tropp and Anna C Gilbert. Signal recovery from random measurements via orthogonal matching pursuit. *IEEE Transactions on information theory*, 53(12):4655–4666, 2007.

Gaurav Verma, Yashi Gupta, Abid M Malik, and Barbara Chapman. Performance evaluation of deep learning compilers for edge inference. In *2021 IEEE international parallel and distributed processing symposium workshops (IPDPSW)*, pp. 858–865. IEEE, 2021.

Zishen Wan, Nandhini Chandramoorthy, Karthik Swaminathan, Pin-Yu Chen, Vijay Janapa Reddi, and Arijit Raychowdhury. Berry: Bit error robustness for energy-efficient reinforcement learning-based autonomous systems. In *2023 60th ACM/IEEE Design Automation Conference (DAC)*, pp. 1–6. IEEE, 2023.

Jialai Wang, Yuxiao Wu, Weiye Xu, Yating Huang, Chao Zhang, Zongpeng Li, Mingwei Xu, and Zhenkai Liang. Your scale factors are my weapon: Targeted bit-flip attacks on vision transformers via scale factor manipulation. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 20103–20112. IEEE, 2025.

Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. Smoothquant: Accurate and efficient post-training quantization for large language models. In *International conference on machine learning*, pp. 38087–38099. PMLR, 2023.

Fan Yao, Adnan Siraj Rakin, and Deliang Fan. {DeepHammer}: Depleting the intelligence of deep neural networks through targeted chain of bit flips. In *29th USENIX Security Symposium (USENIX Security 20)*, pp. 1463–1480, 2020.

Anlan Yu, Shusen Jing, Ning Lyu, Wujie Wen, and Zhiyuan Yan. Error correction output codes for robust neural networks against weight-errors: A neural tangent kernel point of view. *Advances in Neural Information Processing Systems*, 37:82493–82513, 2024.

APPENDIX

A BACKGROUND AND RELATED WORK

Numerical precision for neural deployment. Floating-point and integer-quantized formats represent numerical values differently. A normal binary floating-point value is $x_{\text{fp}} = (-1)^s 2^{e-b}(1+m)$, where s is the sign bit, e the stored exponent, b the exponent bias, and $m \in [0, 1)$ the fractional part. Uniform integer quantization represents $x_{\text{int}} = \Delta(q - q_0)$, where q is a bounded integer code, $\Delta > 0$ a shared scale, and q_0 a zero point. Floating point provides nonuniform spacing across a wide numerical range, which can improve accuracy for distributions with substantial outliers, as demonstrated by FP8 post-training quantization (Kuzmin et al., 2022). Retaining existing floating-point weights also avoids additional integer quantization error and quantizer calibration or fine-tuning (Esser et al., 2020), while supporting established mobile-CPU and edge-GPU inference (Cai et al., 2023). While floating point offers these deployment benefits, its exponent makes weight protection more challenging: a bit flip can drastically change 2^{e-b} or produce a nonfinite value (Hong et al., 2019). Existing defenses largely focus on integer weight codes, whose numerical range remains bounded when their scale and zero point are intact, leaving a substantial gap in protection for floating-point deployment. Integer quantization nevertheless often retains floating-point scales, whose corruption can affect an entire weight group. Flip-S demonstrates that attacking these scales enables effective targeted attacks on quantized models (Wang et al., 2025). Defenses evaluated on integer weight-bit faults, such as RandBET (Stutz et al., 2021), leave scale corruption outside their evaluated scope; protecting weight codes alone therefore does not establish protection of the complete quantized representation. LODISS directly addresses floating-point weight corruption without introducing shared dequantization scales.

Random weight-bit faults. Models deployed on edge devices encounter hardware faults that are absent from idealized software execution. Memory retention failures, reduced supply voltage, and radiation exposure can corrupt stored weights (Liu et al., 2013; Stutz et al., 2021; Rech, 2024; Wan et al., 2023; Putra et al., 2022), while device aging can further weaken memory stability (Gebregiorgis et al., 2016). Such bit flips can substantially alter stored weight values, degrading the model's predictive performance (Hong et al., 2019). Prior studies model random weight faults by independently flipping bits according to a Bernoulli distribution, with the flip probability set by the bit-error rate (Stutz et al., 2021). However, evaluating a single round of bit flips does not capture fault accumulation during long-term deployment. Without repair or reloading, corrupted weights persist, allowing subsequent faults to further degrade predictions and potentially cause system failure. LODISS addresses this accumulation through repeated weight recovery, evaluating subsequent faults on the actual repaired state, including any remaining reconstruction error.

Adversarial weight-bit faults. Edge deployment also exposes models to adversaries capable of inducing hardware faults through techniques such as Rowhammer-based memory corruption (Yao et al., 2020) and laser fault injection during neural-weight reads (Dumont et al., 2023). At the software level, bit-flip attacks identify critical weight bits whose modification can substantially degrade model behavior (Hong et al., 2019; Rakin et al., 2019). Progressive bit search uses gradients to propose candidates and prediction-loss evaluations to select damaging flips (Rakin et al., 2019), while DeepHammer demonstrates attacks using physically realizable memory faults (Yao et al., 2020). Physical injection requires preparation, including memory profiling or spatial and temporal alignment, making attacks that succeed with few targeted flips particularly relevant. Our study focuses on persistent stored-weight corruption and additionally evaluates adaptive attacks that select faults according to their effect after recovery. This tests whether weight repair remains effective when the adversary explicitly targets the recovery mechanism.

Fault-tolerance methods. RandBET combines quantization, clipping, and random weight-bit errors during training (Stutz et al., 2021), while WCAT learns quantization ranges to improve resilience to random and adversarial flips (Chitsaz et al., 2023). Both are developed for integer-quantized weights, leaving their protection unestablished for floating-point representations. OCM-based methods introduce redundancy through separated class codewords (Özdenizci & Legenstein, 2022). Their mechanism does not inherently require integer weights, but the original evaluations use quantized models, and class-code outputs limit direct application to continuous predictions. More fundamentally, these methods preserve predictions without reconstructing damaged parameters,

allowing persistent faults to accumulate beyond the model's tolerance. LoDiSS instead repairs the stored floating-point weights, supporting repeated recovery without classification-specific output codes.

Fault recovery for neural weights. In-place ECC reserves parity bits within constrained integer weight representations, enabling correction without additional weight-storage space (Guan et al., 2019). However, its representation-specific protection has been demonstrated only on INT8-quantized CNNs, without support for floating-point weight protection or non-CNN architectures (Guan et al., 2019). Real-ECC extends numerical recovery to neural weights by projecting full-dimensional variables onto linear constraints and estimating sparse corrections through linear programming (Li et al., 2026). Although promising, this direct adaptation leaves substantial training and recovery overhead: redundant variables enlarge parameter, gradient, and optimizer-state arrays, while recovery optimizes corrections across entire weight groups. Test-time constraint regeneration additionally requires dense QR factorization, delaying repairs on resource-constrained devices. Recovery reliability is also limited by insufficient consideration of floating-point storage: exact consistency can mistake ordinary rounding for corruption and modify intact weights, while nonfinite values caused by bit flips can prevent recovery. Its selected-layer additive-error experiments do not establish effectiveness under repeated floating-point bit faults. LoDiSS addresses these limitations through compact subspace learning, shared seeded transformations that regenerate constraints without QR, and calibrated iterative recovery that handles rounding and nonfinite values while reconstructing only suspected weights. Our comparison uses noise-tolerant Real-ECC recovery with matched representation dimensions, so the measured improvements extend beyond replacing its original exact-consistency requirement.

Real-valued coding and frames. In signal processing, real-valued coding and frames represent signals through redundant linear measurements that support recovery from corrupted or missing values. Candes & Tao (2005) establish sparse-error correction through linear programming, while Candes & Randall (2008) extend recovery to small background errors, including quantization noise. Frame theory explains when the remaining measurements permit reconstruction and how measurement errors affect its accuracy (Fickus & Mixon, 2012). These methods assume that relatively few measurements are corrupted; the underlying signal can remain dense. LoDiSS reinterprets these principles as learning recoverable floating-point weights in seeded subspaces. Normalized orthogonal matching pursuit identifies suspected weight corruptions (Tropp & Gilbert, 2007), which are then reconstructed from the remaining weight values. A global recovery threshold accounts for floating-point rounding errors in clean stored weights, while nonfinite corrupted values, such as NaN and infinity, are treated as missing observations. Only selected weights are rewritten, preserving all other stored entries during repeated repair.

Low-dimensional learning and seeded structure. Parameter prediction from smaller representations (Denil et al., 2013) and optimization within fixed random subspaces (Li et al., 2018) establish that training can use fewer variables than the deployed model. FrameQuant also applies frame representations to neural weights, targeting low-bit transformer quantization (Adepu et al., 2024). These approaches study compact parameterizations or quantization rather than recovery after weight corruption. LoDiSS learns in low-dimensional seeded subspaces while retaining redundancy in the exported weights for subsequent repair. Under matched constraints, this preserves the admissible weight set while reducing parameter, gradient, and optimizer-state storage. Sharing pseudorandom transformations across a layer's weight rows and repeated repairs reduces recovery setup costs, while the seed allows regeneration independently of learned weights. Together with a global threshold calibrated to floating-point rounding errors, this design supports efficient training and reliable, task-agnostic recovery throughout deployment.

B Additional Experimental Results

Does LoDiSS remain robust across floating-point precisions? We evaluate FP16 and FP8-E4M3FN weight storage using ResNet-18 on all four classification datasets. Training includes the target-format weight cast, while master parameters and tensor computations remain FP32. LoDiSS achieves the highest robust accuracy in every dataset–format setting at BERs 10^{-5} and 10^{-4} (Table 4). At 10^{-5}, the average gains over Real-ECC are approximately 67 pp for FP16 and 16 pp for FP8-E4M3FN; at 10^{-4}, both gains exceed 81 pp. After ten recovery cycles at 10^{-6} and 10^{-5},

LoDiSS retains 99.9–100.0% of its initial accuracy to the displayed precision across all 16 conditions (Figure 4). This extends the recovery results beyond FP32 and supports the numerical treatment of stored weights in Challenge 3.

Does LoDiSS withstand multiple adversarial flips between repairs? DeepHammer reports roughly a day of offline memory profiling, followed by 66–96 seconds of online execution to induce 13–20 selected weight-bit flips in its VGG-16 attacks (Yao et al., 2020), while LoDiSS achieves a mean recovery time of approximately 4 seconds for VGG-11 (Figure 3). Our main evaluation therefore assumes a more capable attacker than this hardware-constrained demonstration, allowing all 16 targeted flips to accumulate before recovery without physical injection delays or restrictions on flippable locations. To evaluate long-term fault robustness, we consider an idealized attacker that induces $n \in \{1, 2, 4, 8\}$ selected bit flips simultaneously before each repair across repeated attack–recovery cycles, without physical injection delays or restrictions to hardware-verified flippable locations. Ten attack–recovery cycles introduce up to 80 flips, with each new attack acting on the actual repaired weights. Figure 5 shows higher final accuracy for LoDiSS than for None and Real-ECC across the displayed datasets, architectures, and group sizes. Larger groups reduce accuracy in several models, especially at eight flips per cycle, while many trajectories remain stable. These results demonstrate sustained robustness to long-term adversarial bit flips through repeated recovery, even when multiple weight bits are targeted before each repair.

How does the representation dimension affect recovery? Table 5 compares $\rho \in \{0.3, 0.5, 0.8\}$ and isolates the contribution of recovery. Increasing ρ from 0.3 to 0.8 raises accuracy after ten injections at BER 10^{-5} by 0.5–6.2 pp across the five architectures. For every ratio and both BERs, the final mean accuracy is within 0.1 pp of its pre-injection value at the displayed precision. Thus, the larger representation improves task performance while retaining effective recovery under these faults. At $\rho = 0.8$, enabling recovery adds approximately 76 pp of final accuracy on average over the matched no-recovery condition at 10^{-5}. The separate per-repair gain measures the increase produced by the last repair within a recovery-enabled trajectory. Together, these comparisons support $\rho = 0.8$ as an effective choice among the tested ratios and establish that active repair is responsible for sustained robustness.

How does robustness change as the adversarial budget increases? Figure 6 extends Table 2 to budgets from 1 to 16 flips. At each budget, recovery is evaluated on a copy while the attack continues from the unrepaired state. Averaged across all 20 settings, LoDiSS achieves higher robust accuracy than every evaluated baseline at every budget. Its mean advantage over Real-ECC ranges from approximately 32–64 pp under BFA and 53–77 pp under adaptive BFA. Recovery-aware attacks cause larger reductions in some settings, including CIFAR-100/ResNet-18 and GTSRB/ResNet-34, while other trajectories remain stable. The curves show that the benefit of recovery extends across the tested budgets, including attacks that explicitly account for the repair procedure.

C Theoretical Analysis

We establish how LoDiSS preserves the admissible weight set with less training state, regenerates its recovery structure, and reconstructs corrupted weights. We then relate the recovery conditions to random bit-error rates (BERs) and adversarial bit budgets. All statements concern the layerwise representation and recovery procedure in Section 3; algebraic identities use exact arithmetic, with storage and computation errors included explicitly in Subsection C.6.

C.1 Equal expressiveness with fewer trainable variables

Fix the same recovery constraints and row grouping for both parameterizations. Let F_ℓ have orthonormal columns spanning all solutions of $H_\ell w = 0$, so $P_\ell = \mathrm{I}_{D_\ell} - H_\ell^\top H_\ell = F_\ell F_\ell^\top$. The ambient parameterization in Equation 5 is $W_\ell = V_\ell P_\ell$, whereas LoDiSS learns $W_\ell = Z_\ell F_\ell^\top$ in Equation 7. This comparison isolates the benefit of learning only the independent variables within a fixed recovery structure.

Table 4: **Robust accuracy after one random injection with reduced-precision weights.** ResNet-18 accuracy (%) is reported as mean$_{\pm \mathrm{SD}}$ over three fault realizations from one training run per setting, after recovery where applicable. Column headers specify the storage format and BER. Real-ECC and LoDiSS use $\rho = 0.8$. Bold marks the highest displayed mean within each dataset–format–BER comparison.

Dataset	Method	FP16			FP8-E4M3FN		
		10^{-6}	10^{-5}	10^{-4}	10^{-6}	10^{-5}	10^{-4}
CIFAR-10	None	$9.6_{\pm 0.7}$	$10.2_{\pm 0.8}$	$10.5_{\pm 0.9}$	$89.7_{\pm 0.8}$	$62.2_{\pm 19.6}$	$11.7_{\pm 2.4}$
	OCM-Ozdenizci	$9.3_{\pm 1.5}$	$10.1_{\pm 0.1}$	$9.9_{\pm 0.2}$	$89.7_{\pm 1.0}$	$49.0_{\pm 6.8}$	$10.0_{\pm 0.1}$
	OCM-Hadamard	$10.0_{\pm 0.0}$	$10.2_{\pm 0.4}$	$10.2_{\pm 0.1}$	$86.8_{\pm 5.8}$	$55.0_{\pm 10.7}$	$10.6_{\pm 0.6}$
	Real-ECC	$\mathbf{90.7_{\pm 0.0}}$	$13.7_{\pm 3.9}$	$9.7_{\pm 0.2}$	$89.3_{\pm 1.9}$	$84.6_{\pm 5.7}$	$10.3_{\pm 0.3}$
	LoDiSS (Ours)	$90.3_{\pm 0.0}$	$\mathbf{90.3_{\pm 0.0}}$	$\mathbf{90.3_{\pm 0.0}}$	$\mathbf{90.0_{\pm 0.0}}$	$\mathbf{90.0_{\pm 0.0}}$	$\mathbf{90.0_{\pm 0.1}}$
CIFAR-100	None	$0.9_{\pm 0.1}$	$1.0_{\pm 0.1}$	$1.0_{\pm 0.0}$	$64.2_{\pm 1.2}$	$12.1_{\pm 3.4}$	$0.9_{\pm 0.1}$
	OCM-Ozdenizci	$1.0_{\pm 0.0}$	$1.0_{\pm 0.0}$	$1.0_{\pm 0.1}$	$57.6_{\pm 8.3}$	$9.5_{\pm 8.9}$	$1.1_{\pm 0.1}$
	OCM-Hadamard	$1.0_{\pm 0.1}$	$0.9_{\pm 0.1}$	$0.9_{\pm 0.0}$	$56.5_{\pm 10.7}$	$9.5_{\pm 3.6}$	$0.9_{\pm 0.1}$
	Real-ECC	$\mathbf{65.6_{\pm 0.0}}$	$4.6_{\pm 4.1}$	$1.0_{\pm 0.0}$	$\mathbf{64.9_{\pm 0.5}}$	$26.8_{\pm 13.6}$	$1.0_{\pm 0.3}$
	LoDiSS (Ours)	$64.0_{\pm 0.0}$	$\mathbf{64.0_{\pm 0.0}}$	$\mathbf{63.9_{\pm 0.0}}$	$64.0_{\pm 0.0}$	$\mathbf{64.0_{\pm 0.0}}$	$\mathbf{64.0_{\pm 0.1}}$
SVHN	None	$9.5_{\pm 1.6}$	$10.9_{\pm 4.4}$	$7.4_{\pm 1.2}$	$95.2_{\pm 0.7}$	$87.4_{\pm 3.8}$	$13.9_{\pm 4.1}$
	OCM-Ozdenizci	$11.3_{\pm 7.2}$	$11.0_{\pm 6.2}$	$9.6_{\pm 2.8}$	$95.9_{\pm 0.9}$	$87.9_{\pm 6.3}$	$14.3_{\pm 0.4}$
	OCM-Hadamard	$9.5_{\pm 1.5}$	$13.7_{\pm 4.4}$	$8.2_{\pm 1.9}$	$\mathbf{96.4_{\pm 0.3}}$	$83.2_{\pm 15.7}$	$15.9_{\pm 2.6}$
	Real-ECC	$\mathbf{96.0_{\pm 0.0}}$	$38.3_{\pm 50.0}$	$9.8_{\pm 2.7}$	$95.7_{\pm 0.4}$	$92.8_{\pm 2.2}$	$9.2_{\pm 0.0}$
	LoDiSS (Ours)	$95.9_{\pm 0.0}$	$\mathbf{95.9_{\pm 0.0}}$	$\mathbf{95.9_{\pm 0.0}}$	$96.1_{\pm 0.0}$	$\mathbf{96.1_{\pm 0.0}}$	$\mathbf{96.1_{\pm 0.0}}$
GTSRB	None	$1.8_{\pm 1.2}$	$2.4_{\pm 0.6}$	$1.7_{\pm 1.1}$	$98.0_{\pm 0.1}$	$64.2_{\pm 34.3}$	$3.3_{\pm 0.7}$
	OCM-Ozdenizci	$1.9_{\pm 0.4}$	$2.9_{\pm 1.4}$	$2.6_{\pm 2.0}$	$97.6_{\pm 0.8}$	$74.0_{\pm 10.1}$	$1.9_{\pm 1.2}$
	OCM-Hadamard	$2.9_{\pm 2.7}$	$4.3_{\pm 2.1}$	$1.5_{\pm 1.2}$	$98.0_{\pm 0.4}$	$78.4_{\pm 26.5}$	$2.6_{\pm 0.6}$
	Real-ECC	$\mathbf{98.1_{\pm 0.0}}$	$24.9_{\pm 41.6}$	$2.3_{\pm 0.5}$	$\mathbf{98.2_{\pm 0.0}}$	$79.4_{\pm 31.9}$	$0.9_{\pm 0.4}$
	LoDiSS (Ours)	$98.0_{\pm 0.0}$	$\mathbf{98.0_{\pm 0.0}}$	$\mathbf{98.0_{\pm 0.0}}$	$97.9_{\pm 0.0}$	$\mathbf{97.9_{\pm 0.0}}$	$\mathbf{97.8_{\pm 0.1}}$

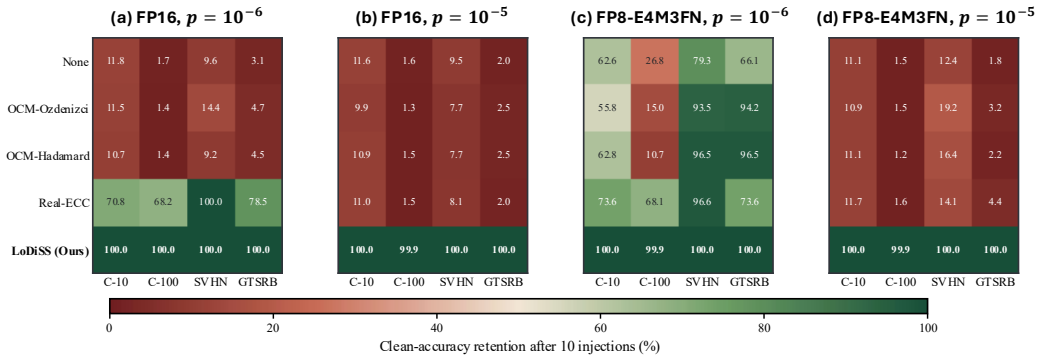


Figure 4: **Accuracy retention after ten random injection–recovery cycles.** From left to right, panels show FP16 at BERs 10^{-6} and 10^{-5}, followed by FP8-E4M3FN at the same rates. Cells give mean retention (%) over three ResNet-18 fault trajectories, normalized by each model's own pre-injection accuracy. Each repair is retained before the next injection.

Proposition 1 (Equal expressiveness within a fixed subspace)**.** *The parameterizations $Z_\ell F_\ell^\top$ and $V_\ell P_\ell$ represent the same weight matrices. For any target matrix $W_{\ell,\star} \in \mathbb{R}^{n_\ell \times D_\ell}$,*

$$\begin{aligned} \min_{Z_\ell} \|W_{\ell,\star} - Z_\ell F_\ell^\top\|_{\mathrm{F}}^2 &= \min_{V_\ell} \|W_{\ell,\star} - V_\ell P_\ell\|_{\mathrm{F}}^2 \\ &= \|W_{\ell,\star} N_\ell\|_{\mathrm{F}}^2, \end{aligned} \tag{18}$$

where $\|\cdot\|_{\mathrm{F}}$ is the Frobenius norm. The unique minimizer over Z_ℓ is $Z_{\ell,\star} = W_{\ell,\star} F_\ell$.

Proof. For any V_ℓ, set $Z_\ell = V_\ell F_\ell$ to obtain $Z_\ell F_\ell^\top = V_\ell P_\ell$. Conversely, $V_\ell = Z_\ell F_\ell^\top$ satisfies $V_\ell P_\ell = Z_\ell F_\ell^\top$. For the approximation error, decompose

$$W_{\ell,\star} - Z_\ell F_\ell^\top = W_{\ell,\star} N_\ell + (W_{\ell,\star} F_\ell - Z_\ell) F_\ell^\top. \tag{19}$$

Since $N_\ell F_\ell = 0$, these terms are orthogonal in the Frobenius inner product. Since $F_\ell^\top F_\ell = \mathrm{I}_{d_\ell}$, the squared error equals $\|W_{\ell,\star} N_\ell\|_{\mathrm{F}}^2 + \|W_{\ell,\star} F_\ell - Z_\ell\|_{\mathrm{F}}^2$, giving the stated minimizer. □

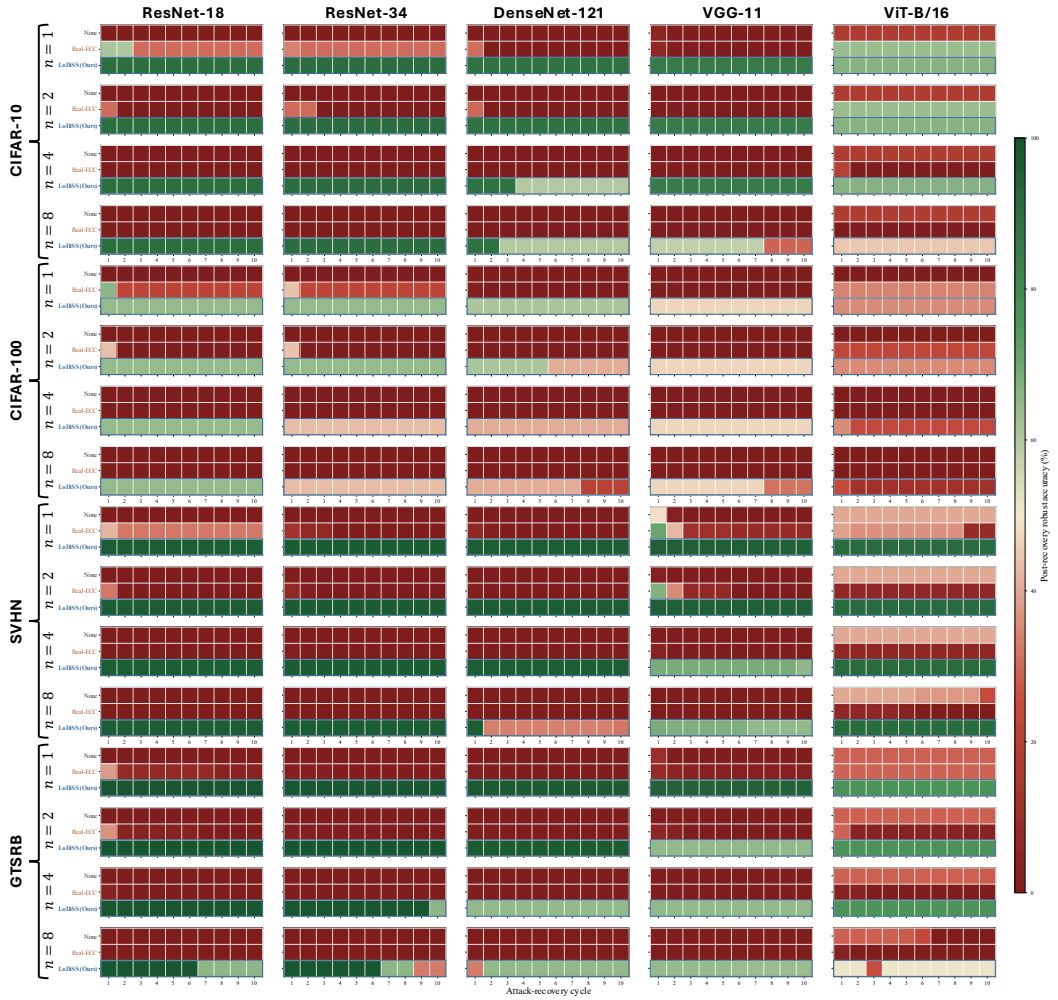


Figure 5: **Robust accuracy over ten grouped BFA–recovery cycles.** Each cycle applies $n \in \{1, 2, 4, 8\}$ flips selected using pre-recovery loss, followed by one committed repair. Panel rows group datasets and values of n; panel columns show architectures. Within each panel, columns show cycles 1–10 and rows show None, Real-ECC, and LoDiSS. Colors give mean FP32 accuracy (%) over three paired attack-calibration splits. OCM-based methods were not evaluated in this experiment.

Thus direct subspace learning removes redundant variables without adding approximation error within the chosen recovery constraints. It also preserves the attainable task-loss values when the remaining network is fixed. Under ordinary gradient descent, a step $Z_\ell^+ = Z_\ell - \eta_{\mathrm{opt}}(\nabla_{W_\ell}\mathcal{L})F_\ell$ induces

$$W_\ell^+ = W_\ell - \eta_{\mathrm{opt}}(\nabla_{W_\ell}\mathcal{L})P_\ell, \tag{20}$$

which is the same weight update as ordinary gradient descent on V_ℓ from matching initial weights in exact arithmetic. Here $\eta_{\mathrm{opt}} > 0$ is the learning rate. Thus the compact representation preserves both the admissible weight set and ordinary gradient updates under matched constraints. In particular, any target satisfying $W_{\ell,\star}N_\ell = 0$ is represented exactly.

The initialization in Equation 7 preserves the average variance of the source weights. Conditional on the fixed frame F_ℓ, suppose the transposed initial weight row $w_{\ell,a}^{\mathrm{init}}$ has zero mean and covariance $(2/D_\ell)\mathrm{I}_{D_\ell}$. Its initialized representation $z_{\ell,a}^{(0)} = \sqrt{D_\ell/d_\ell}\, F_\ell^\top w_{\ell,a}^{\mathrm{init}}$ then has covariance $(2/d_\ell)\mathrm{I}_{d_\ell}$.

Table 5: **Representation-dimension and recovery ablations.** Entries report accuracy (%), except per-repair gain, which is in pp. Before-injection values are mean$_{\pm SD}$ across four datasets. Faulted values summarize three trajectories after ten injections, with each trajectory first averaged across datasets. The no-recovery condition starts from the same $\rho = 0.8$ learned weights and uses matched fault sequences. Per-repair gain compares accuracy immediately before and after the tenth repair within the $\rho = 0.8$ recovery-enabled trajectory. Dashes indicate inapplicable entries. Bold marks the highest displayed mean among the three representation dimensions.

Architecture	Evaluation	LoDiSS with recovery			$\rho = 0.8$	Per-repair
		$\rho = 0.3$	$\rho = 0.5$	$\rho = 0.8$	no recovery	gain (pp)
ResNet-18	Before injection	$82.5_{\pm 17.6}$	$84.8_{\pm 17.3}$	$\mathbf{86.6_{\pm 15.8}}$	—	—
	BER 10^{-6}	$82.5_{\pm 0.0}$	$84.8_{\pm 0.0}$	$\mathbf{86.6_{\pm 0.0}}$	$4.5_{\pm 0.0}$	$80.2_{\pm 1.8}$
	BER 10^{-5}	$82.5_{\pm 0.0}$	$84.9_{\pm 0.0}$	$\mathbf{86.6_{\pm 0.0}}$	$4.5_{\pm 0.0}$	$82.0_{\pm 0.0}$
ResNet-34	Before injection	$82.7_{\pm 18.6}$	$85.2_{\pm 16.7}$	$\mathbf{86.9_{\pm 15.6}}$	—	—
	BER 10^{-6}	$82.7_{\pm 0.0}$	$85.2_{\pm 0.0}$	$\mathbf{86.9_{\pm 0.0}}$	$4.5_{\pm 0.0}$	$82.4_{\pm 0.0}$
	BER 10^{-5}	$82.7_{\pm 0.0}$	$85.2_{\pm 0.0}$	$\mathbf{86.9_{\pm 0.0}}$	$4.5_{\pm 0.0}$	$82.4_{\pm 0.0}$
DenseNet-121	Before injection	$79.0_{\pm 18.5}$	$81.9_{\pm 18.0}$	$\mathbf{85.2_{\pm 16.8}}$	—	—
	BER 10^{-6}	$79.0_{\pm 0.0}$	$81.9_{\pm 0.0}$	$\mathbf{85.2_{\pm 0.0}}$	$4.5_{\pm 0.0}$	$78.1_{\pm 1.9}$
	BER 10^{-5}	$79.0_{\pm 0.0}$	$81.9_{\pm 0.0}$	$\mathbf{85.2_{\pm 0.0}}$	$4.5_{\pm 0.0}$	$80.6_{\pm 0.0}$
VGG-11	Before injection	$77.0_{\pm 22.1}$	$79.2_{\pm 22.7}$	$\mathbf{79.8_{\pm 22.7}}$	—	—
	BER 10^{-6}	$77.0_{\pm 0.0}$	$79.2_{\pm 0.0}$	$\mathbf{79.8_{\pm 0.0}}$	$4.5_{\pm 0.0}$	$72.2_{\pm 4.0}$
	BER 10^{-5}	$77.0_{\pm 0.0}$	$79.2_{\pm 0.0}$	$\mathbf{79.9_{\pm 0.0}}$	$4.5_{\pm 0.0}$	$75.3_{\pm 0.0}$
ViT-B/16	Before injection	$65.8_{\pm 23.2}$	$65.9_{\pm 23.2}$	$\mathbf{66.3_{\pm 23.5}}$	—	—
	BER 10^{-6}	$65.8_{\pm 0.0}$	$65.9_{\pm 0.0}$	$\mathbf{66.3_{\pm 0.0}}$	$5.3_{\pm 0.7}$	$59.9_{\pm 0.0}$
	BER 10^{-5}	$65.8_{\pm 0.0}$	$65.9_{\pm 0.0}$	$\mathbf{66.3_{\pm 0.0}}$	$4.5_{\pm 0.0}$	$61.3_{\pm 0.6}$

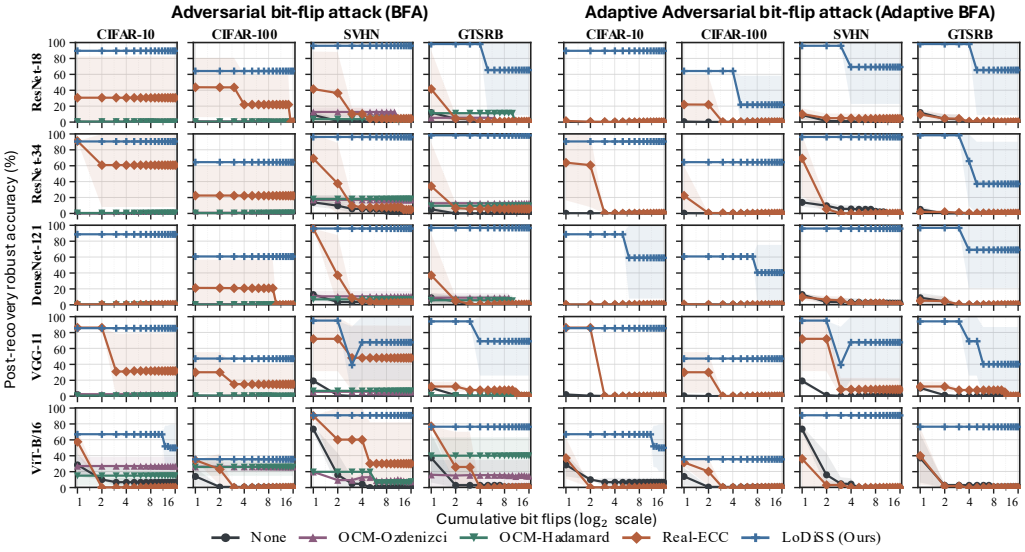


Figure 6: **Robust accuracy as the adversarial bit-flip budget increases.** Rows show architectures, and each half shows the four datasets. The left half reports BFA; the right half reports adaptive BFA (Adapt.), matching Table 2. Lines and shaded bands show mean FP32 accuracy and standard deviation over three paired attack-calibration splits. Repairs are evaluated on copies and do not change the attack state. The horizontal axis uses a base-2 logarithmic scale.

Consequently,

$$\mathrm{Var}(\varphi_{\ell,i}^{\top} z_{\ell,a}^{(0)}) = \frac{2}{d_\ell}\|\varphi_{\ell,i}\|_2^2,$$
$$\frac{1}{D_\ell}\sum_{i=1}^{D_\ell} \mathrm{Var}(\varphi_{\ell,i}^{\top} z_{\ell,a}^{(0)}) = \frac{2}{D_\ell}. \tag{21}$$

This follows from covariance propagation and $\sum_i \|\varphi_{\ell,i}\|_2^2 = d_\ell$. Under the stated isotropy assumption, the rescaling maintains the fan-in variance convention while introducing recovery redundancy (He et al., 2015).

C.2 Training memory savings

Addressing Challenge 1, direct subspace learning replaces the $n_\ell D_\ell$ trainable entries of V_ℓ with the $n_\ell d_\ell$ entries of Z_ℓ without further restricting W_ℓ. Consider an optimizer with a fixed amount of state per trainable entry. Let β_{state} be the bytes used by one parameter value and its initialized optimizer state, and let β_{grad} be the bytes per gradient value. With the same optimizer and scalar storage choices, the persistent saving is $\beta_{\text{state}} \sum_{\ell=1}^{L} n_\ell(D_\ell - d_\ell)$ bytes. Resident gradients add $\beta_{\text{grad}} \sum_{\ell=1}^{L} n_\ell(D_\ell - d_\ell)$ bytes. This separates the reduction in trainable values from the precision chosen for each state array.

To relate this saving to total memory, let M_{sub} and M_{full} denote peak allocated memory over intervals with parameters and optimizer state resident. Let $M_{\text{sub}}^{\text{other}}$ and $M_{\text{full}}^{\text{other}}$ denote the respective peaks of all other allocations. Then

$$M_{\text{full}} - M_{\text{sub}} = \beta_{\text{state}} \sum_{\ell=1}^{L} n_\ell(D_\ell - d_\ell) + M_{\text{full}}^{\text{other}} - M_{\text{sub}}^{\text{other}}. \tag{22}$$

The remaining allocations include expanded weights, gradients, activations, saved tensors for backpropagation, and optimizer workspace. This decomposition identifies the persistent training-state reduction and the condition under which it lowers peak memory: the saved state exceeds any additional temporary storage.

C.3 Regenerating seeded subspaces efficiently

The frame in Equation 10 combines permutations, pair rotations, and signs, each of which is orthogonal. Their product therefore satisfies

$$F_\ell^\top F_\ell = E_\ell^\top E_\ell = \mathrm{I}_{d_\ell}.$$

For example, each pair rotation has the form

$$\begin{pmatrix} \cos\omega & \sin\omega \\ -\sin\omega & \cos\omega \end{pmatrix}, \tag{23}$$

where ω is its seeded angle; an unpaired entry is unchanged. Orthogonality is supplied by construction, rather than by factorizing a sampled dense matrix. The fixed seed and layer index specify the subspace $\mathcal{S}_\ell$ independently of the learned representations. Regenerating the same factors reproduces F_ℓ and the checks N_ℓ used for recovery. This is why the seed can replace a stored dense basis without replacing the task information carried by the weights.

We count calls to tensor-library operations, including random-array generation, indexing, and elementwise arithmetic. In each mixing round, one call generates a permutation tensor and another generates an angle tensor. A constant number of elementwise calls computes their rotation coefficients, and one sign tensor is shared across the construction. Factor initialization therefore uses $O(T_\ell)$ tensor calls. Applying F_ℓ to all n_ℓ rows in a batch also uses $O(T_\ell)$ calls for padding, permutations, pairwise rotations, and signs. Its adjoint reverses the transformations and retains the first d_ℓ entries. Since $T_\ell = O(\log D_\ell)$, both call counts grow logarithmically with row width and remain independent of the number of output rows. The generation tensors contain $O(D_\ell)$ entries per round, while application operates on $n_\ell \times D_\ell$ arrays.

The construction supplies orthogonality directly without QR factorization. One initialization serves all weight rows, and subsequent repairs reuse the cached factors, amortizing generation across the layer and repeated recovery intervals. The tensor-library call counts describe this batched procedure, while the runtime measurements in Figure 3 evaluate its efficiency on the tested hardware.

Cached permutations, rotation coefficients, and signs occupy $O(T_\ell D_\ell)$ entries, independently of n_ℓ, compared with $D_\ell d_\ell$ entries for an explicit frame. In addition to factor initialization, pursuit normalization transiently materializes a $d_\ell \times D_\ell$ frame before retaining only length-D_ℓ norm arrays.

Recovery setup accounts for this computation and its temporary storage. Applying the factors to all n_ℓ rows with reusable buffers requires $O(n_\ell D_\ell + T_\ell D_\ell)$ workspace, including the cached factors; backpropagation also retains the intermediate tensors it needs. Shared factor storage complements the training-state saving from learning Z_ℓ in Subsection C.2. Moreover, $\mathrm{rank}(W_\ell) \le \min(n_\ell, d_\ell)$, so choosing $d_\ell \ge n_\ell$ permits full row rank while retaining recovery constraints within each row.

C.4 Reconstructing weights from intact entries

The recovery procedure addresses Challenge 1 through selected-support reconstruction and Challenge 3 through calibrated localization and numerical checks. Consider row a of layer ℓ, with the working variables y, r, J, U defined in the method. Its consistency residual carries the same information as the conventional syndrome: $N_\ell y = H_\ell^\top H_\ell y$ and $H_\ell y = H_\ell N_\ell y$. Both follow from $H_\ell^\top H_\ell = N_\ell$ and $H_\ell H_\ell^\top = \mathrm{I}_{D_\ell - d_\ell}$, allowing the checks to be evaluated without constructing H_ℓ.

For the ith standard basis vector $e^{(i)} \in \mathbb{R}^{D_\ell}$, define the syndrome atom $\gamma_{\ell,i} = N_\ell e^{(i)}$. Since $N_\ell^2 = N_\ell = N_\ell^\top$, its norm satisfies $\|\gamma_{\ell,i}\|_2^2 = 1 - (P_\ell)_{ii}$ and its correlation with any residual $r \in \mathrm{range}(N_\ell)$ is $\gamma_{\ell,i}^\top r = r_i$. Thus Equation 14 ranks normalized atom correlations, following orthogonal matching pursuit (Tropp & Gilbert, 2007). The threshold and progress test determine whether an entry is added. If nonfinite observations initialize the support, the first score uses the zero-filled residual r_0 before refitting.

For a selected support J, let $A_{\ell,J} = (N_\ell)_{:,J}$ collect its atoms. Then $A_{\ell,J}^\top A_{\ell,J} = M_{\ell,J}$ and $A_{\ell,J}^\top r_0 = (r_0)_J$, so Equation 15 solves the normal equations of $\min_h \|r_0 - A_{\ell,J} h\|_2^2$. After localization, the same small matrix supports reconstruction from the retained entries. The following result characterizes reconstruction and its sensitivity to numerical error when the selected support contains the faults and the retained rows span the representation space.

Proposition 2 (Recovery from the retained entries)**.** *$M_{\ell,J}$ is invertible if and only if $F_{\ell,U}$ has full column rank. Under this condition, Equation 17 computes $\widehat{z} = F_{\ell,U}^\dagger y_U$. If J contains every corrupted entry and $y_U = F_{\ell,U} z_{\ell,a} + \eta_U$, then*

$$\|\widehat{z} - z_{\ell,a}\|_2 \le \frac{\|\eta_U\|_2}{\sigma_{\min}(F_{\ell,U})}, \tag{24}$$

where $\sigma_{\min}$ is the smallest singular value. The representation $z_{\ell,a}$ and ideal selected weights $F_{\ell,J} z_{\ell,a}$ are recovered exactly when $\eta_U = 0$, independently of the corrupted values on J.

Proof. The Parseval identity gives $F_{\ell,U}^\top F_{\ell,U} = \mathrm{I}_{d_\ell} - F_{\ell,J}^\top F_{\ell,J}$. The determinant identity for the products $F_{\ell,J}^\top F_{\ell,J}$ and $F_{\ell,J} F_{\ell,J}^\top$ makes its invertibility equivalent to that of $M_{\ell,J}$. The Woodbury identity then gives

$$(F_{\ell,U}^\top F_{\ell,U})^{-1} = \mathrm{I}_{d_\ell} + F_{\ell,J}^\top M_{\ell,J}^{-1} F_{\ell,J}. \tag{25}$$

Multiplication by $F_{\ell,U}^\top y_U$ yields Equation 17, identifying it with retained-row least squares. Hence $\widehat{z} - z_{\ell,a} = F_{\ell,U}^\dagger \eta_U$, and $\|F_{\ell,U}^\dagger\|_2 = 1/\sigma_{\min}(F_{\ell,U})$ proves the bound. □

The selected values also satisfy $\widehat{w}_J = \xi$: multiplying $\widehat{z} = g + F_{\ell,J}^\top \xi$ by $F_{\ell,J}$ and using $M_{\ell,J}\xi = F_{\ell,J} g$ proves the identity. Thus an $|J|$-dimensional system implements retained-row least squares without solving a d_ℓ-dimensional system. For fixed J, the recovered values depend only on y_U, not on the magnitude or sign of the discarded values. This directly accommodates large exponent errors and known nonfinite erasures once their positions are included in J.

Corollary 1 (A sufficient condition for stable reconstruction)**.** *For a nonempty support J, let $\chi_{\ell,J} = \|F_{\ell,J}\|_2^2$. If $\chi_{\ell,J} < 1$, then*

$$\sigma_{\min}(F_{\ell,U}) = \sqrt{1 - \chi_{\ell,J}}, \qquad \mathrm{cond}_2(M_{\ell,J}) \le \frac{1}{1 - \chi_{\ell,J}}. \tag{26}$$

Consequently, $\chi_{\ell,J} \le 1 - \kappa_{\max}^{-1}$ suffices for the exact support system to pass the configured condition-number limit.

Proof. Both $F_{\ell,U}^\top F_{\ell,U} = \mathrm{I} - F_{\ell,J}^\top F_{\ell,J}$ and $M_{\ell,J} = \mathrm{I} - F_{\ell,J} F_{\ell,J}^\top$ have smallest eigenvalue $1 - \chi_{\ell,J}$; the largest eigenvalue of $M_{\ell,J}$ is at most one. □

This condition relates stable recovery to the actual selected frame rows (Fickus & Mixon, 2012). At most $D_\ell - d_\ell$ erasures can be removed while retaining full column rank. The implemented condition-number check controls relative conditioning; the explicit $\chi_{\ell,J}$ condition additionally bounds absolute error amplification. These results establish reconstruction once localization supplies a suitable support.

C.5 Computational cost of iterative recovery

For an active row of layer ℓ, let K_p count support refits, including a rejected final trial, and let s be the largest support considered. The initial check and refits use $1 + K_\mathrm{p}$ syndrome evaluations, each applying $F_\ell^\top$ and F_ℓ. Each refit constructs selected frame rows, forms a Gram matrix of order at most s, and invokes numerical least squares. The final reconstruction adds a condition-number calculation and, when accepted, a linear solve with the products in Equation 17. Selected rows are obtained by applying the cached frame adjoint to standard basis vectors, without generating new random factors.

Batching these vectors makes each selected-row construction one adjoint application. The resulting procedure uses $O((K_\mathrm{p}+1)T_\ell)$ structured tensor operations and $O(K_\mathrm{p}+1)$ matrix-product, solver, and reduction invocations. These are operation-invocation counts; their operands grow with s, d_ℓ, and D_ℓ. Constructing selected rows sequentially instead uses up to s adjoint applications per refit and permits $O(T_\ell D_\ell + D_\ell + s d_\ell + s^2)$ workspace after norm initialization. Batched construction can additionally require an $s \times D_\ell$ buffer. The small systems scale with the selected support, whereas Equation 6 optimizes corrections across the protected group. These expressions account for workspace and operand sizes; the recovery times in Figure 3 also reflect iterations, conditioning, and batching.

C.6 Preserving floating-point weights across repairs

For the finite clean stored row $\bar{w}_{\ell,a}$, write $\bar{w}_{\ell,a} = F_\ell z_{\ell,a} + \eta$, where η includes evaluation and storage-rounding errors relative to the exact expansion. The row quantities $\tau_{\ell,a}^{\mathrm{clean}}$ in Equation 11 are deterministic functions of the clean weights and specified recovery arithmetic. They are computed at export or verified-clean loading to form τ_{global} in Equation 12; only this single calibration scalar is retained and protected throughout deployment. If checking an unchanged clean row reproduces its calibrated residual, then $\|r_0\|_\infty \le \tau_{\ell,a}^{\mathrm{clean}} \le \tau_{\mathrm{global}}$, so recovery skips it on every check. Thus global calibration preserves the stored bits across repeated fault-free checks despite ordinary numerical residuals.

Assume standard binary floating-point storage, round-to-nearest rounding, gradual underflow, and no overflow. For recovery of row (ℓ, a), let q_J denote the reconstructed values immediately before the final rounding to the stored-weight precision. When storage and recovery precisions coincide, this can be the rounding in the final arithmetic operation. Let $\varepsilon_{\mathrm{rec}} \ge 0$ bound all preceding numerical errors, including frame regeneration and the linear solve, so that

$$\|q_J - F_{\ell,J} F_{\ell,U}^\dagger y_U\|_\infty \le \varepsilon_{\mathrm{rec}}.$$

Let v_J^{new} denote the resulting stored values. The final rounding is the one already performed when producing these outputs.

Corollary 2 (Restoring the original stored weights)**.** *Suppose an accepted reconstruction of row (ℓ, a) uses a nonempty set J containing every corrupted position, and $F_{\ell,U}$ has full column rank. Let $h_{\ell,J} = \max_{i \in J} \|\varphi_{\ell,i}\|_2$. Define $m_{\ell,J}$ as the minimum, over $i \in J$, of the distance from the original stored value $(\bar{w}_{\ell,a})_i$ to a boundary of its floating-point numerical rounding interval. If*

$$\mathcal{E}_{\ell,J} := h_{\ell,J} \frac{\|\eta_U\|_2}{\sigma_{\min}(F_{\ell,U})} + \varepsilon_{\mathrm{rec}} + \|\eta_J\|_\infty < m_{\ell,J}, \tag{27}$$

the selected outputs restore the original numerical values. They also restore the original bits if every restored zero has its original sign. All unselected bits remain unchanged.

Proof. Since U contains only clean observations, $y_U = F_{\ell,U} z_{\ell,a} + \eta_U$. By Proposition 2 and Cauchy–Schwarz,

$$\|F_{\ell,J} F_{\ell,U}^{\dagger} y_U - F_{\ell,J} z_{\ell,a}\|_\infty \le h_{\ell,J} \frac{\|\eta_U\|_2}{\sigma_{\min}(F_{\ell,U})}.$$

The definition of $\varepsilon_{\mathrm{rec}}$ and $[\bar{w}_{\ell,a}]_J = F_{\ell,J} z_{\ell,a} + \eta_J$ therefore give $\|q_J - [\bar{w}_{\ell,a}]_J\|_\infty \le \mathcal{E}_{\ell,J}$. The strict margin condition places each reconstructed value inside the rounding interval of its original stored value, so final rounding reproduces that numerical value. In the assumed binary representation, finite nonzero values have unique bit encodings; zero additionally requires the stated sign condition. □

The same analysis also bounds the error of approximate restoration. Let u_{store} be the unit roundoff of the stored-weight precision and let λ_{store} be its smallest positive subnormal value. The stored outputs satisfy

$$\|v_J^{\mathrm{new}} - [\bar{w}_{\ell,a}]_J\|_\infty \le \mathcal{E}_{\ell,J} + u_{\mathrm{store}} \|q_J\|_\infty + \lambda_{\mathrm{store}}/2.$$

The bound separates error amplification through the retained frame, numerical recovery error, and final storage rounding. Together with clean calibration, it explains how the recovery procedure accommodates floating-point weights, following the principle of allowing small background errors in numerical recovery (Candes & Randall, 2008).

If every affected row at each interval through K satisfies Corollary 2, including the zero-sign condition, and unaffected rows remain unchanged, then induction gives $X^{(t)} = X^{(0)}$ for every $t \in \{1, \dots, K\}$ in Equation 2. Each successful interval restores the clean bits before subsequent faults arrive, preventing their accumulation across intervals. For approximate repairs, the remaining error enters the retained-observation term in the next reconstruction bound. This analysis follows the evolving stored weights used in the repeated-repair evaluation.

Connection to the efficiency and robustness results. At a common fraction $\rho = d_\ell / D_\ell$, the parameter, gradient, and per-parameter optimizer arrays for the selected layers occupy a fraction ρ of their ambient counterparts, explaining the approximately 20% weight-state saving at $\rho = 0.8$ in Figure 3. Shared generation and selected-support solves provide complementary mechanisms for reducing training and recovery overhead, with implementation efficiency measured in that figure. Restoring the stored bits also preserves the network's outputs under the same inference computation for every task, since other model state remains intact under Section 2. This task-independent mechanism supports the design of weight-level recovery, with empirical robustness assessed in Table 1, Figure 2, and Table 3. The comparison with noise-tolerant ℓ_1 recovery in Real-ECC evaluates the complete recovery procedures under the same fault protocols.

C.7 When iterative localization finds the corrupted entries

The preceding reconstruction results apply after support selection. We next give a sufficient condition for the actual pursuit rule, including its threshold, progress test, and condition-number gate. This result isolates finite sparse errors in exact arithmetic; Subsection C.6 accounts separately for floating-point reconstruction. For layer ℓ, define normalized syndrome atoms and their coherence by

$$a_{\ell,i} = \sqrt{(N_\ell)_{ii}}, \qquad \widehat{\gamma}_{\ell,i} = \frac{N_\ell e^{(i)}}{a_{\ell,i}}, \qquad \mu_\ell = \max_{i \ne j} |\widehat{\gamma}_{\ell,i}^{\top} \widehat{\gamma}_{\ell,j}|. \tag{28}$$

Assume $a_{\ell,i}^2 \ge \epsilon_{\mathrm{fp}} > 0$ for every entry, so the normalization in Equation 14 equals the atom norm. Write $a_\ell^{\min} = \min_i a_{\ell,i}$ and $a_\ell^{\max} = \max_i a_{\ell,i}$.

Proposition 3 (Sufficient conditions for successful pursuit)**.** *Let $y = F_\ell z_{\ell,a} + e$ be finite, with support $J_\star = \mathrm{supp}(e)$ of size $1 \le s \le D_\ell - d_\ell$ and no clean numerical error. Define $c_{\min} = \min_{i \in J_\star} a_{\ell,i} |e_i|$, $\lambda_s = 1 - (s-1)\mu_\ell$, and $\Lambda_s = 1 + (s-1)\mu_\ell$. With $\tau = \tau_{\mathrm{global}}$ from Equation 12, suppose*

$$\begin{aligned} (2s-1)\mu_\ell < 1, \qquad & \frac{a_\ell^{\min} \lambda_s c_{\min}}{\sqrt{s}} > 1.5\tau, \\ \delta \le 1 - \sqrt{1 - \lambda_s/s}, \qquad & \kappa_{\max} \ge \left(\frac{a_\ell^{\max}}{a_\ell^{\min}} \right)^2 \frac{\Lambda_s}{\lambda_s}. \end{aligned} \tag{29}$$

Then exact evaluation of Algorithm 2 selects precisely $J_\star$, passes its reconstruction gate, and restores the clean row.

Proof. Let G collect the normalized atoms indexed by $J_\star$. Their Gram matrix has eigenvalues in $[\lambda_s, \Lambda_s]$. For an atom outside $J_\star$, diagonal dominance gives $\|G^\dagger \widehat{\gamma}_{\ell,j}\|_1 \le s\mu_\ell/\lambda_s < 1$. Thus its correlation with any nonzero $r \in \mathrm{range}(G)$ is strictly smaller than $\|G^\top r\|_\infty$. Exact refitting makes the correlations of already selected atoms zero, so each new selection lies in $J_\star$. Moreover,

$$\|G^\top r\|_\infty \ge \sqrt{\lambda_s/s}\,\|r\|_2, \qquad \|r\|_2 \ge \sqrt{\lambda_s}\,c_{\min}$$

until all true positions are selected. The second inequality follows by minimizing over selected coefficients: at least one unselected coefficient retains magnitude $c_{\min}$. The winning unnormalized entry therefore exceeds 1.5τ, which bounds every adaptive threshold. Adding its normalized atom and refitting removes at least $(\lambda_s/s)\|r\|_2^2$ of squared residual, since the orthogonal component of a unit atom has norm at most one. The stated bound on δ ensures acceptance. Finally, write $a_{\ell,J_\star} = (a_{\ell,i})_{i\in J_\star}$. Then $M_{\ell,J_\star} = \mathrm{diag}(a_{\ell,J_\star}) G^\top G\, \mathrm{diag}(a_{\ell,J_\star})$ satisfies the stated condition-number bound. After all s selections the residual is zero, and Proposition 2 gives exact reconstruction. □

For finite errors with $E = \varnothing$, this sufficient result connects successful pursuit to three properties: separated syndrome directions, detectable error amplitudes, and stable reconstruction. It applies to any corrupted positions satisfying the stated inequalities, including adversarial selections. Nonfinite values supply known erasure positions through $J = E$, and Proposition 2 establishes reconstruction once the final support contains both these erasures and any finite errors.

C.8 Random bit-error rates and repeated recovery

The relevant random-fault load is the number of affected weights in each recovery row. Let b_{fp} be the number of eligible bits per stored weight. Under Section 2, a word is touched in one injection with probability

$$q(p) = 1 - (1-p)^{b_{\mathrm{fp}}} \le b_{\mathrm{fp}} p, \qquad S_{\ell,a}^{(t)} \sim \mathrm{Binomial}(D_\ell, q(p)), \tag{30}$$

where $S_{\ell,a}^{(t)}$ counts touched words in row (ℓ, a) at interval t. For the product probability below, the rows occupy disjoint stored words; tied weight arrays are counted once. This count includes finite and nonfinite changes and upper-bounds the number of numerically changed or nonfinite entries; changing the sign of zero, for example, changes bits without changing the numerical value.

Proposition 4 (A BER range controlling support overload)**.** *Choose an integer support allowance $0 \le s_\ell < D_\ell$ for each layer, and define its one-injection overload probability*

$$\pi_\ell(p) = \sum_{j=s_\ell+1}^{D_\ell} \binom{D_\ell}{j} q(p)^j [1-q(p)]^{D_\ell - j}. \tag{31}$$

Across K independent injections, the probability that every row stays within its allowance is

$$\Pr\Big[S_{\ell,a}^{(t)} \le s_\ell \text{ for all } \ell, a, t\Big] = \prod_{\ell=1}^{L} [1-\pi_\ell(p)]^{K n_\ell} \ge 1 - K \sum_{\ell=1}^{L} n_\ell \pi_\ell(p). \tag{32}$$

In particular, for a target overload probability $0 < \varepsilon < 1$, the sufficient BER range is specified by

$$K \sum_{\ell=1}^{L} n_\ell \binom{D_\ell}{s_\ell + 1} (b_{\mathrm{fp}} p)^{s_\ell+1} \le \varepsilon. \tag{33}$$

Proof. Distinct stored words and injection intervals use disjoint independent Bernoulli trials, giving the binomial law and product. An overloaded row contains a subset of $s_\ell + 1$ touched words. A union bound over such subsets gives $\pi_\ell(p) \le \binom{D_\ell}{s_\ell+1} q(p)^{s_\ell+1}$. Union-bounding rows and intervals and using $q(p) \le b_{\mathrm{fp}} p$ proves Equation 33. □

For N_{row} rows with common width D and allowance s, this gives the explicit range

$$p \le \frac{1}{b_{\mathrm{fp}}} \left[\frac{\varepsilon}{K N_{\mathrm{row}} \binom{D}{s+1}} \right]^{1/(s+1)}. \tag{34}$$

The exact binomial tails give a tighter range. When every pattern within the chosen support allowance satisfies the localization, conditioning, and floating-point restoration conditions, these probabilities lower-bound the probability of successful recovery across the K intervals. The bounds use the actual recovery-row width D_ℓ and the independent fault masks in Section 2. They connect the per-interval BER and repair frequency to the support size that each row needs to accommodate.

Why repair limits accumulation. Without recovery, a bit differs from its original value after K injections with probability $p_K = [1-(1-2p)^K]/2$, obtained by counting an odd number of toggles. A stored word consequently differs with probability $1-(1-p_K)^{b_{\mathrm{fp}}}$. When each interval restores the original bits, every new interval instead faces the one-injection load $q(p)$; deployment duration enters through Equation 32. This explains the benefit of repeated repair without treating the entire lifetime's flips as one unrecovered burst. Inexact repairs follow the retained-error analysis in Subsection C.6.

Relation to the evaluated BERs. For the FP32 experiments, $p = 10^{-6}, 10^{-5}, 10^{-4}$ corresponds to approximately 0.00320%, 0.0320%, and 0.3195% of weight words touched per injection. This sparse fault load supports localizing corrupted entries and solving small reconstruction systems. The binomial tails quantify how the per-row load grows with BER, while Table 1 and Figure 2 evaluate recovery at these rates. The sustained performance across ten repair cycles is consistent with the benefit of correcting faults before their effects accumulate.

C.9 Adversarial bit budgets and detectable corruption

To analyze adversarial placement, including concentration of all flips in one row, define the minimum nonzero support in the seeded subspace by

$$\Delta_\ell = \min_{z \neq 0} \|F_\ell z\|_0, \tag{35}$$

where $\|\cdot\|_0$ counts nonzero entries. This quantity measures how many weight values must change to move between distinct valid vectors.

Proposition 5 (Error, erasure, and invisible-attack budgets)**.** *In exact arithmetic, a valid row is uniquely determined from an observation with at most s unknown errors and r known erasures whenever $2s + r < \Delta_\ell$. Any nonzero numerical change e invisible to the exact checks, $N_\ell e = 0$, modifies at least Δ_ℓ weights and therefore requires at least Δ_ℓ bit flips. If every d_ℓ rows of F_ℓ are independent, then $\Delta_\ell = D_\ell - d_\ell + 1$.*

Proof. Two valid explanations of the same observations differ at no more than $2s+r$ positions. Their difference lies in $\mathrm{range}(F_\ell)$, contradicting Equation 35 unless they coincide. Likewise, $N_\ell e = 0$ places a nonzero change in that range, and each altered numerical weight requires at least one altered bit. Under the row-independence assumption, a nonzero $F_\ell z$ cannot vanish at d_ℓ positions, giving support at least $D_\ell - d_\ell + 1$. Conversely, a nonzero vector orthogonal to any $d_\ell - 1$ rows gives support at most this value. □

A B-bit attack changes at most B weight entries. Therefore $2B < \min_\ell \Delta_\ell$ guarantees a unique exact explanation even if all flips target one row, with Δ_ℓ determined by the actual seeded frame. Under the stated row-independence assumption, the ideal unique-correction radius is $B \leq \min_\ell \lfloor (D_\ell - d_\ell)/2 \rfloor$. With a common dimension fraction ρ, each row has $(1-\rho)D_\ell$ checks, relating representation redundancy to this ideal correction capacity. These identifiability results apply to both ordinary and adaptive adversaries; Proposition 3 supplies sufficient conditions for the implemented pursuit to find the unique solution in the finite-error setting.

Detection of large floating-point changes. For $1 \leq B \leq D_\ell$, define

$$\alpha_\ell(B) = \min_{1 \leq |J| \leq B} \sqrt{\lambda_{\min}((N_\ell)_{J,J})}. \tag{36}$$

For every finite B-sparse error, idempotence gives $\|N_\ell e\|_2^2 = e^\top N_\ell e \geq \alpha_\ell(B)^2 \|e\|_2^2$. Let $\varepsilon_{\mathrm{check}}$ bound the infinity-norm error of the computed residual relative to $N_\ell y$. With the fixed global threshold, the row is detected whenever

$$\frac{\alpha_\ell(B)\|e\|_2}{\sqrt{D_\ell}} > \tau_{\mathrm{global}} + \|N_\ell \bar{w}_{\ell,a}\|_\infty + \varepsilon_{\mathrm{check}}. \tag{37}$$

This follows from the reverse triangle inequality. Moreover, Equation 28 gives $\alpha_\ell(B) \geq a_\ell^{\min}\sqrt{1-(B-1)\mu_\ell}$ when the radicand is positive. The condition identifies when finite exponent errors remain detectable despite adversarial placement under the global calibrated threshold. Nonfinite errors activate recovery directly through the erasure set, giving complementary detection paths for floating-point corruption.

C.10 Recovery-aware attacks and their effect after repair

The implemented adaptive objective. The objective in Equation 1 differs from ordinary BFA by placing recovery inside the loss. Let $\mathcal{R}(v)$ be numerical weight recovery with all retained recovery information fixed, where v concatenates the working weight arrays. Backward pass differentiable approximation (Athalye et al., 2018) supplies proposal gradients through the replacement

$$\frac{\partial \mathcal{R}(v)}{\partial v} \longleftarrow \mathrm{I}, \qquad \widetilde{\nabla}_v \mathcal{L}_{\mathcal{C}}(\mathcal{R}(v)) = \nabla_w \mathcal{L}_{\mathcal{C}}(w)|_{w=\mathcal{R}(v)}\,. \tag{38}$$

Here $\mathcal{L}_{\mathcal{C}}(w)$ denotes the same mean prediction loss evaluated directly on numerical weights. The identity approximation supplies gradients for proposing candidate weights, whose discrete bit flips are then evaluated explicitly. Let M_k contain the k previously selected flips, let $\mathcal{Q}_k$ be the candidate bit positions proposed from these gradients, and let $\mathbf{e}_j^{\mathrm{bit}}$ toggle bit j. Greedy selection evaluates

$$\begin{aligned} j_k &\in \operatorname*{arg\,max}_{j \in \mathcal{Q}_k \setminus \mathrm{supp}(M_k)} \mathcal{L}_{\mathcal{C}}\big(\mathsf{Repair}(X \oplus M_k \oplus \mathbf{e}_j^{\mathrm{bit}})\big)\,, \\ M_{k+1} &= M_k \oplus \mathbf{e}_{j_k}^{\mathrm{bit}}, \qquad M_0 = 0. \end{aligned} \tag{39}$$

The candidate set searches all bits of the high-gradient weights specified in Subsection 4.1, including flips producing nonfinite values. Each score uses actual recovery on a temporary model copy, so final ranking does not rely on the identity approximation. In fixed-budget experiments, X is the clean attack starting state and recovery is not committed during search. With repeated repairs, each interval starts from the previous committed state and applies the selected burst before the next repair.

Corollary 3 (Restoration removes the effect of a selected attack)**.** *Suppose an attack produces supports for which recovery selects every changed position and satisfies Corollary 2, including its zero-sign condition. If all unaffected rows remain unchanged, then*

$$\mathsf{Repair}(X^{(0)} \oplus M) = X^{(0)}, \qquad \mathcal{L}_{\mathcal{C}}\Big(\mathsf{Repair}(X^{(0)} \oplus M)\Big) = \mathcal{L}_{\mathcal{C}}(X^{(0)}). \tag{40}$$

This holds regardless of whether the attack selected M using pre-recovery or post-recovery loss.

Proof. Selected-support reconstruction uses only the retained weights, so it is independent of corrupted magnitudes on the selected support. The storage-restoration condition returns the original values and bits, while selective writeback preserves every other bit. The resulting network and loss are therefore unchanged. □

The corollary shows that successful weight restoration removes an attack's effect on the loss even when fault selection explicitly accounts for recovery. Proposition 5 relates exactly check-invisible changes to the bit budget, Proposition 3 gives sufficient conditions for finite-error localization, and Corollary 2 establishes restoration of the stored values. Together, these results explain how recovery redundancy, selective reconstruction, and floating-point calibration support resistance to targeted faults. The fixed-budget results in Table 2 evaluate both BFA and adaptive BFA, while the repeated-burst experiments in Subsection 4.2 assess recovery as adversarial damage accumulates between repairs.