SANAI: BIT-FLIP ATTACK-RESISTANT SECURE NEURAL NETWORK WITH EMBEDDED AND KEYED PARITY

Anonymous authors
Paper under double-blind review

ABSTRACT

Quantized neural networks store weights in dense memory, where bit flips caused by voltage scaling, aging, radiation, or deliberate attacks such as RowHammer can severely degrade model behavior. A single corrupted high-order bit can lead to erroneous classifications or nonsensical language-model outputs. Embedded error-correcting codes (ECC) avoid dedicated parity storage by hiding parity within model weights. However, existing schemes use largely uniform embedding across layers and bit planes, leave parity unsecured, and tolerate only low bit-error rates (BERs), typically up to 6%. Moreover, an adversary aware of sensitive weight locations and the ECC scheme can bypass protection by deliberately exceeding the correction capability of targeted codewords. We propose SANAI, a sensitivity-guided, bit-plane-selective scheme that robustly protects critical model weights against bit flips with zero additional parity-storage overhead. SANAI secures ECC parity through keyed embedding across a two-dimensional weight space. Spread-transform dither modulation binds parity to a secret key, while keyed interleaving disperses concentrated attacks across codewords. We derive a survival law showing that robustness is governed by the correctable fraction t/n, rather than correction strength t alone, and establish a closed-form bound on the protectable fraction of a model. Implemented as a streaming checker on an 8×8 systolic array, SANAI verifies weights on the load path with only 0.41% area overhead and no additional memory. Across ten vision backbones, three quantization schemes, and ten language models ranging from 82M to 13B parameters, SANAI maintains accuracy within approximately one percentage point of the clean baseline at BERs up to 12%, twice the maximum BER supported by prior embedded-ECC approaches. It also survives fine-tuning bit-exactly, neutralizes progressive and targeted bit-flip attacks, and exactly corrects AttentionBreaker and SilentStriker attacks.

1 INTRODUCTION

Deep networks increasingly run on energy-constrained accelerators (Jouppi et al., 2017), where weights quantized to eight bits or fewer sit in dense memory. Reduced-voltage operation buys efficiency and pays in bit errors (Chandramoorthy et al., 2019; Koppula et al., 2019; Zhang et al., 2018; Stutz et al., 2021); aging and radiation add more (Reagen et al., 2018; Baumann, 2005), memory errors are ordinary in deployed fleets (Schroeder et al., 2011), and RowHammer lets an adversary produce them on purpose (Kim et al., 2014; Yao et al., 2020). Because a quantized weight packs its value into few bits, one high-order flip in a sensitive layer can collapse the whole model (Hong et al., 2019; Rakin et al., 2019).

Uniform ECC guards every stored bit at a standing storage and energy price, although most bits barely move the loss. Embedded schemes such as E3-CODE (Kamineni et al., 2026) fold check bits into the weights themselves, but apply one policy to every layer and bit plane, leave the parity readable by anyone who holds the weights, and stop at a six percent BER. We instead concentrate the budget on the few fragile layers and damaging bits, hide the parity at zero added memory in bits the robust layers do not need, bind it to a key, and carry resilience well past six percent.

SANAI (Sensitivity-Aware Neural-network protection using Adaptive cross-layer STDM embeddIng, Figure 1) identifies sensitive layers by direct fault injection, protects only their most significant bit planes with a BCH code, and embeds the parity into low-order bits of robust carrier layers

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.

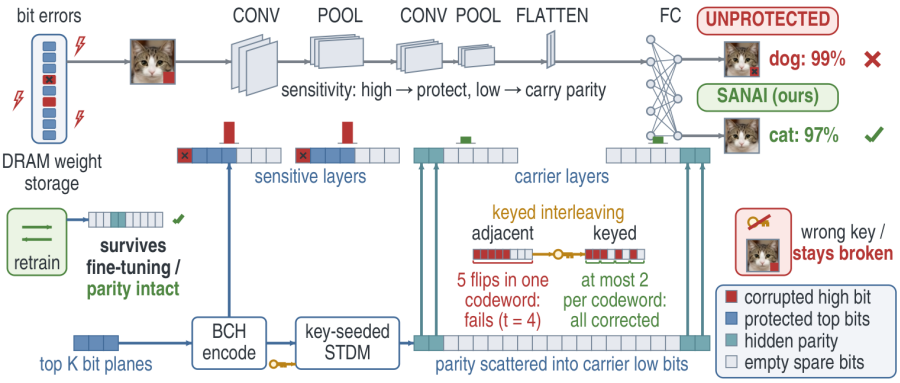


Figure 1: SANAI overview. Bit errors in weight memory flip high-order bits and the unprotected model misclassifies (top). SANAI probes layer sensitivity (bars), BCH-encodes the top K bit planes of the sensitive few, assigns weights to codewords by a keyed permutation, and hides the parity with key-seeded STDM in the low-order bits of robust carrier layers at zero added memory (bottom). Inset: the same attack flips overload one codeword under adjacent grouping but stay within t under keyed interleaving. The parity survives fine-tuning and is unreadable without the key.

using spread-transform dither modulation (STDM) seeded by a secret key. At read time the key regenerates the projection and BCH decoding repairs the high-order bits; a keyed permutation decides which weights share a codeword, so concentrated flips cannot overload any codeword. Because recovery is a pure read-path operation, the construction admits a streaming hardware checker on the weight bus of a systolic accelerator.

Contributions. (i) A sensitivity metric and $O(L \log L)$ selection rule that replaces the integer program of prior work on embedded ECC with a sensitivity-based sorting step needing no solver, retraining, or training data. (ii) A survival law (Proposition 1) showing that t/n, not t, governs robustness, and a closed-form ceiling on the protectable fraction of a model at zero memory. (iii) A key-seeded STDM embedding that is parity carrier and access control at once (Proposition 2), and keyed interleaving with a balls-into-bins guarantee (Theorem 1). (iv) A custom hardware realization: the embedding is exactly a keystream-whitened LSB write (Proposition 3); synthesized on an 8×8 systolic array, it detects every single-bit flip, including the sign bit, at 0.41% area and zero added memory. (v) An evaluation across ten vision backbones, three quantizers, and ten LLMs against random errors, PBS, T-BFA, AttentionBreaker, and SilentStriker.

2 Background and Related Work

Fault tolerance for neural networks. Conventional ECC keeps extra check bits and corrects reliably at a large memory and energy cost (Reagen et al., 2018). Activation clipping and error-aware retraining (Hong et al., 2019; He et al., 2020) harden the model rather than the stored weights, need training data, and cannot recover the original parameters; selective protection (Koppula et al., 2019) leans on retraining. SANAI protects stored weights directly with no added memory and no retraining.

Embedded weight ECC. Weight nulling (Qin et al., 2017), opportunistic parity (Burel et al., 2021), value-aware parity (Lee & Yang, 2022), and E3-CODE (Kamineni et al., 2026) conceal parity inside the weight bits, but apply one policy to every layer and bit plane, keep parity next to the data it protects, and stop at 6% BER. SANAI differs on all three counts: placement follows a measured sensitivity ranking, protection is confined to the high-order planes of the fragile few, and the parity moves cross-layer into tolerant carriers (Table 10, Appendix D).

Bit-flip attacks and detection. Progressive bit search (PBS) (Rakin et al., 2019), targeted attacks (Rakin et al., 2020; 2021), and, on LLMs, AttentionBreaker (Das et al., 2024) and SilentStriker (Xu et al., 2025) show that a handful of chosen flips suffices. Detection defenses (Li et al., 2021; Javaheripi & Koushanfar, 2021; Liu et al., 2023) raise an alarm rather than repair.

Data hiding. Quantization index modulation (Chen & Wornell, 2001) and its spread-transform variant (Malvar & Florêncio, 2003) embed information into a host by quantization, and weight

watermarking (Uchida et al., 2017; Adi et al., 2018; Darvish Rouhani et al., 2019) embeds ownership signatures; we repurpose STDM as a key-seeded carrier for functional parity.

3 METHOD

3.1 PROBLEM FORMULATION AND FAULT MODEL

A trained network has quantized weight tensors $\theta = \{\theta_\ell\}_{\ell=1}^{L}$ with $W = \sum_\ell |\theta_\ell|$ weights at b bits each, so the stored bitstring $s(\theta)$ has Wb bits. Weights are quantized per output channel, $q = \mathrm{clip}(\mathrm{round}(w/s), -2^{b-1}, 2^{b-1}-1)$ with $s = \max|w|/(2^{b-1}-1)$, in two's complement, so a flip in bit plane j perturbs the integer by 2^j and a sign flip moves the weight by half its range. Memory is a cause-agnostic binary symmetric channel $\mathcal{F}_p$ flipping each stored bit independently with probability p, which is also the substrate for the burst protocol of prior work and the adversarial attackers of Sections 5.3 and 5.4. We adopt a localized fault model in which errors land in the protected layers' memory regions while carrier regions stay clean (revisited in Section 7).

The defender applies an offline encoder $E : \theta \mapsto \theta'$ and a keyed read-time decoder D_κ. Writing $A(\cdot)$ for task accuracy, the design problem is

$$\max_{E,\, D_\kappa} \ \mathbb{E}_{\mathcal{F}_p}\big[A\big(D_\kappa(\mathcal{F}_p(E(\theta)))\big)\big] \tag{1}$$

subject to (i) zero added memory, $|s(\theta')| = |s(\theta)|$; (ii) no retraining, so E may not move a weight beyond its quantization step; and (iii) key-gated recovery, so for any wrong key the decoder output is no better than the unprotected corrupted model. SANAI exploits a measured double asymmetry. Across layers, damage is concentrated: scoring every weight of ResNet-18/CIFAR-10 by its attack-damage proxy puts the damage in the top 1% of weights, and protecting the layers that hold them (24.8% of the model) keeps 95.0% at 6% BER, while extending protection to the next 8% of weights doubles coverage and parity for no gain (Master Table 4, rows V1.34 and V1.35); the remaining 91% is the carrier pool. Within a weight, the damage of a flip decays geometrically with bit position.

3.2 SENSITIVITY-GUIDED LAYER SELECTION

For each layer ℓ we inject flips at a small probe rate p_0 into that layer alone and record the accuracy drop

$$S_\ell \ = \ A_{\mathrm{clean}} - \mathbb{E}_{\epsilon\sim\mathrm{Bern}(p_0)}\big[A(\theta_\ell \oplus \epsilon)\big]. \tag{2}$$

Layers are sorted by S_ℓ and the protected set $\mathcal{P}$ is grown greedily, most fragile first, to a coverage target ρ:

$$\mathcal{P} = \{\ell_{(1)}, \ldots, \ell_{(j)}\}, \qquad j = \max\Big\{j : \textstyle\sum_{i\le j} |\theta_{\ell_{(i)}}| \le \rho W\Big\}. \tag{3}$$

The remaining layers form the carrier set $\mathcal{C}$, least sensitive first. In every network we studied a few early or narrow layers are fragile but hold few weights, while the large late layers are robust and hold most parameters, so the fragile layers are cheap to protect and the robust ones supply carrier bits. For LLM attacks we score layers with a gradient-informed probe $\sum |g \odot w|$, since magnitude is blind to the small attention projections attacks target (Section 5.4); random-BER runs use magnitude. Selection is one sort, $O(L \log L)$ (Appendix B).

3.3 BIT-PLANE BCH ENCODING AND A SURVIVAL LAW

Within each protected layer we encode only the top K bit planes: groups of G weights form GK-bit messages for a binary BCH code (Bose & Ray-Chaudhuri, 1960) over $\mathrm{GF}(2^7)$ ($x^7 + x + 1$, parent length $n = 127$), shortened to the message. The strong code is shortened from the parent $\mathrm{BCH}(127, 50, t{=}12)$ and stores 48 message bits plus 88 parity bits per codeword, so $K = 4$ gives $G = 12$; the light code is shortened from $\mathrm{BCH}(127, 99, t{=}4)$ with 96 message and 32 parity bits, $G = 24$ at $K = 4$. The $G \le t$ property that makes any sign-bit attack correctable (Section 3.5) holds for the $K = 4$, $G = 12$, $t = 12$ configuration used in all LLM experiments, not for $K = 3$ ($G = 16$) or the light code ($G = 24$). The governing quantity for survival is not t but the correctable fraction $f = t/n$ of the parent.

Proposition 1 (Codeword survival). *If the channel flips each of the n codeword bits independently with probability p, the decoder fails exactly when $X \sim \mathrm{Bin}(n,p)$ exceeds t, so $P_{\text{fail}} = \sum_{i=t+1}^{n} \binom{n}{i} p^i (1-p)^{n-i}$, and for any $p < f$, $P_{\text{fail}} \le \exp\big(-2n(f-p)^2\big)$.*

Survival is exponentially reliable whenever $p < f$ (Remark 1 turns this into a design rule). Protecting $K > 1$ planes widens the flat region further, since residual errors below the corrected planes perturb a weight by at most 2^{b-K-1} steps. Under an adversary the relevant quantity is the per-codeword count f_c: a codeword survives iff $f_c \le t$, random faults keep $\mathbb{E}[f_c]$ small, and a targeted attack maximizes $\max_c f_c$ (Appendix C).

3.4 Key-Seeded STDM Parity Hiding

The parity is embedded into the low-order bits of carrier weights by STDM seeded by a secret key κ. To embed one bit β into a group of L_s carrier weights w, the key generates a direction $v \in \{-1,+1\}^{L_s}$ and a dither d; the projection $p = w \cdot v + d$ is snapped to one of two interleaved lattices of step Δ selected by β and written back with the minimum-norm update:

$$p^\star = \Delta\left\lfloor \tfrac{p-d-\beta\Delta/2}{\Delta} \right\rceil + d + \tfrac{\beta\Delta}{2}, \qquad w' = w + \tfrac{p^\star - p}{L_s}\, v, \qquad \hat{\beta} = \left\lfloor \tfrac{w'\cdot v + d}{\Delta/2} \right\rfloor \bmod 2. \tag{4}$$

Each carrier weight moves by at most $\Delta/2$, one low-bit step, so the embedding is invisible in accuracy; it generalizes to B bits per symbol (Appendix B).

Proposition 2 (Recovery and access control). *(a) With the correct key, v and d cancel exactly, so $\hat{\beta} = \beta$ whenever the read-time perturbation of the projection stays below $\Delta/2$. (b) With a wrong key $\kappa' \neq \kappa$ the regenerated direction satisfies $\mathbb{E}[v \cdot v'] = 0$, the projection is uncorrelated with β, each parity bit is recovered with probability $\frac{1}{2}$, and all P_{bits} parity bits are recovered with probability $2^{-P_{\text{bits}}}$, negligible at the operating points of this paper ($P_{\text{bits}} > 10^6$).*

Property (a) is the recovery guarantee and (b) the access-control guarantee: without the key the parity is noise. At read time the key regenerates (v, d), the parity is de-projected, and the BCH decoder corrects the top K planes (Algorithm 1, Appendix B).

3.5 Keyed Codeword Interleaving

In the default layout a codeword covers G adjacent weights, the weak point against a targeted attacker: PBS concentrates flips on sensitive weights that cluster in a few early layers and drive f_c far above t. We break adjacency with a keyed permutation π of each layer's protected indices, assigning weight i to codeword

$$\mathrm{cw}(i) = \lfloor \pi(i)/G \rfloor, \qquad \mathrm{pos}(i) = \pi(i) \bmod G. \tag{5}$$

Adjacent weights land in unrelated codewords, encoder and decoder share the mapping through κ, and no parity or memory is added.

Theorem 1 (Interleaving bound). *Let π be a uniformly random keyed permutation and let the attacker choose F flips without knowledge of π. Over M codewords, $\mathbb{E}[\max_c f_c] = O\big(F/M + \log M/\log\log M\big)$, which is $O(1)$ for $F \ll M \log M$, so with high probability $\max_c f_c \le t$.*

A second guarantee needs no key: a sign-bit attack flips one plane per weight and each weight belongs to one codeword, so it injects at most G errors per codeword, and $G \le t$ makes every sign-bit attack of any size exactly correctable on protected layers (Section 5.4).

3.6 Zero-Memory Feasibility and the Protection Ceiling

The code is systematic, so only the parity, p bits per k_s-bit message ($88/48$ strong, $32/96$ light), must be evicted to carriers. With $W_p = \rho W$ protected weights, replication R ($R = 1$ by default), $W_c = (1-\rho)W$ carriers with h spare host bits each, and STDM at $B \le h$ bits per group of L_s carriers, demand and supply are

$$D = W_p K R\, \tfrac{p}{k_s}, \quad C_{\text{STDM}} = \tfrac{W_c B}{L_s} \le W_c h, \quad D \le C_{\text{STDM}} \iff \rho \le \rho^\star = \tfrac{B k_s}{B k_s + L_s K R p}. \tag{6}$$

The parity cost is $r = \rho K p / k_s$ bits per weight (1.10 predicted versus 1.05 to 1.10 measured, Appendix G), with zero added memory regardless of r, and $\rho^\star \approx 0.48$: at most 48% of a model can

be protected before the parity no longer fits. Above $\rho^\star$ the encoder places the remaining parity in the low-order bits of the protected layers themselves; those bits sit inside the fault region but belong to the codeword and are corrected like any other, so coverage can exceed $\rho^\star$ (52.8% in Section 5.5) until every layer's host bits are exhausted (INT4, Appendix G).

Remark 1 (Choice of t)**.** Proposition 1 and Eq. 6 pull in opposite directions: a larger t widens the safe region $p < t/n$, but each additional correctable error costs roughly m more parity bits, which lowers $\rho^\star$. The right t is the largest one whose parity still fits at the coverage the fragile layers need. At $m = 7$, $t = 12$ gives $f = 9.4\%$ with 88 parity bits and $\rho^\star \approx 48\%$, above the 20 to 40% coverage used here, whereas $t = 18$ would raise f to 14.2% at a parity cost that pushes $\rho^\star$ below that range; enlarging the field never helps, since at $m = 8$ even $t = 20$ gives a smaller f (7.8%) at higher cost.

4 Experimental Setup

Vision. CIFAR-10, CIFAR-100 (Krizhevsky, 2009), and ImageNet-1K (Deng et al., 2009). Ten backbones from 3.5M to 85.8M parameters: ResNet-18/50 and VGG-16 (He et al., 2016; Simonyan & Zisserman, 2015), MobileNet-V2 and EfficientNet-B0 (Sandler et al., 2018; Tan & Le, 2019), Wide ResNet-50-2 (Zagoruyko & Komodakis, 2016), RegNet-Y-16GF (Radosavovic et al., 2020), and Swin-T, MaxViT-T, and ViT-B/16 (Liu et al., 2021; Tu et al., 2022; Dosovitskiy et al., 2021) (sizes in Table 11, Appendix E). We use post-training quantization (PTQ) with per-output-channel scales and two quantization-aware methods, Jacob (Jacob et al., 2018) and LSQ (Esser et al., 2020), at INT8 and INT6 precision.

Language. Ten decoder-only transformers from 82M to 13B parameters: DistilGPT-2 and GPT-2 (Sanh et al., 2019; Radford et al., 2019), Qwen2-0.5B/1.5B, Qwen2.5-7B (Yang et al., 2024a;b), Llama-3.2-1B/3B, Llama-3-8B-Instruct, Llama-3.1-8B (Dubey et al., 2024), and Llama-2-13B (Touvron et al., 2023), at signed INT8 (per-tensor below 7B, per-channel above); embedding and output-head matrices are excluded from injection, protection, and carriers. We report perplexity on WikiText-2/103 (Merity et al., 2017) and 5-shot accuracy on MMLU (Hendrycks et al., 2021), ARC (Clark et al., 2018), HellaSwag (Zellers et al., 2019), and TruthfulQA (Lin et al., 2022) (2,000 questions, three seeds).

Fault models and operating points. *Random BER* flips bits uniformly at a target rate; *burst* injection follows the E3-CODE protocol; *PBS* (Rakin et al., 2019) is gradient-guided and untargeted (500 flips); *T-BFA* (Rakin et al., 2021) forces a source class to a target class, measured by attack success rate (ASR); AttentionBreaker and SilentStriker are re-implemented on SANAI's own INT8 tensors. The mechanism is instantiated at three operating points (Table 12, Appendix E): the strong $t = 12$ code at $K = 3$ ($K = 4$ for depthwise networks and ImageNet backbones, $K = 5$ for ViT-B/16), 20% coverage (40% on ImageNet) for random BER; the light $t = 4$ code at $K = 4$ ($G = 24$), 10% coverage, keyed interleaving for concentrated attacks; and the strong code at $K = 4$ ($G = 12 \leq t$) for LLMs (gradient ranking and keyed interleaving for attacks; magnitude ranking, adjacent grouping under random BER). STDM uses $B = 5$, $L_s = 1$, $R = 1$, and $p_0 = 2\%$ unless noted.

Experiments run on four NVIDIA RTX 6000 Ada GPUs; protecting the largest vision model takes a few minutes, dominated by the sensitivity probes. And on the Hardware Side we evaluated the experiments with Synopsys VCS Tool for verification and Synopsys dc compiler for synthesis on gscl45nm technology node.

5 Results

5.1 Random Bit Errors Across Architectures and Datasets

Table 1 (block 1) reports protected top-1 accuracy; the embedding itself is invisible, since at 0% BER every protected CIFAR checkpoint matches the unprotected INT8 model to two decimals (Master Table 3). On CIFAR-10 the three classic architectures stay within about one point of clean through a twelve percent BER; on CIFAR-100 the residual and plain models hold within one to two points and EfficientNet-B0 through ten percent. Jacob QAT matches PTQ; LSQ's learned scales are more fragile, so a matched budget collapses at 12% (27.4%) while $2.5\times$ the coverage restores 85.8% (rows V1.6,

Table 1: Vision results at INT8 and zero added memory. Block 1: protected top-1 (%) under random BER, strong code ($K = 3$; $K = 4$ for EfficientNet-B0 and ImageNet backbones other than ResNet-18; $K = 5$ for ViT-B/16). Block 2: burst protocol of Kamineni et al. (2026), prior embedded ECC as reported on their reference models versus SANAI on ours, so retention normalized to each method's own clean accuracy is the comparable metric. Block 3: adversarial bit flips, light code, $K = 4$, 10% coverage (T-BFA rows: strong code, high-bin selection). Every row traces to Master Tables 3, 5, and 6.

Block 1. Random BER: protected top-1 (%)

Model	Data	Cov.	1%	3%	6%	8%	10%	12%
ResNet-18	CIFAR-10	19.3%	94.95	94.66	94.54	94.46	94.36	93.06
ResNet-50	CIFAR-10	19.2%	94.87	94.79	94.51	94.54	94.10	93.70
VGG-16	CIFAR-10	20.0%	90.12	89.98	89.87	89.72	89.76	89.19
ResNet-18	CIFAR-100	14.0%	77.30	77.10	76.82	76.69	76.11	75.86
ResNet-50	CIFAR-100	36.1%	75.64	75.36	75.32	75.41	75.36	74.71
EffNet-B0	CIFAR-100	30.0%	63.29	62.49	61.96	61.46	59.97	53.22
ResNet-18	ImageNet	40.4%	76.05	76.14	75.95	75.70	75.78	75.70
ResNet-50	ImageNet	40.9%	82.72	82.52	82.41	82.06	81.31	80.88
MaxViT-T	ImageNet	40.9%	88.14	88.06	88.05	87.97	87.69	87.55
Swin-T	ImageNet	40%	86.50	86.28	85.42	85.34	84.92	83.44
Wide RN-50-2	ImageNet	40%	83.91	83.97	84.06	83.91	83.58	83.58
RegNet-Y-16GF	ImageNet	40%	85.66	85.50	85.05	84.64	84.62	84.48
ViT-B/16	ImageNet	43.5%	85.50	85.42	85.28	84.44	83.25	77.89

Block 2. Burst errors versus prior embedded ECC: top-1 (%) at 1.5 / 3 / 6% BER

Method	ResNet-18 C10	EffNet-B0 C10	ResNet-50 IN	MobileNet-V2 IN
Weight nulling	82.0 / 19.0 / 0.05	82.0 / 2.9 / 0.0	75.9 / 18.8 / 0.4	69.9 / 5.5 / 0.0
Opportunistic parity	81.7 / 81.4 / 10.9	80.3 / 64.8 / 9.8	75.8 / 73.8 / 0.0	65.8 / 58.1 / 0.1
Value-aware parity	82.0 / 81.2 / 77.5	82.0 / 77.9 / 22.6	75.9 / 75.1 / 65.1	69.9 / 66.0 / 20.9
E3-CODE	81.8 / 81.8 / 78.8	82.3 / 78.6 / 55.6	75.9 / 75.8 / 70.0	69.9 / 66.7 / 36.9
E3-CODE-H	82.0 / 81.9 / 81.3	82.7 / 82.3 / 80.1	76.1 / 75.9 / 74.7	71.2 / 70.5 / 65.5
SANAI (ours)	94.9/94.7/**94.5**	83.8/83.0/**81.4**	82.2/81.6/**80.5**	78.4/78.1/**77.8**

Retention at 6% BER, normalized to each method's own baseline over eight settings: 0.01, 0.06, 0.58, 0.72, 0.94, **0.97**.

Block 3a. Progressive bit search, 500 flips, keyed interleaving, light code: top-1 (%)

Model	Data	Cov.	Unprotected	SANAI
ResNet-18 / ResNet-50	CIFAR-10	10%	10.1 / 35.0	**94.9 / 94.8**
ResNet-18 / ResNet-50	CIFAR-100	10%	1.2 / 0.9	**75.5 / 75.8**

Block 3b. Same flipped bits, adjacent / keyed grouping, ResNet-18 / CIFAR-10

Flips	cw $> t$	max/cw	Top-1 (%)
100	3 / 0	12 / 2	66.0 / **95.1**
500	22 / 0	14 / 3	10.1 / **94.9**

Block 3c. Targeted attack (T-BFA): attack success rate (%)

Model	Data, pair	Unprotected	SANAI
ResNet-18	CIFAR-10, $3 \to 5$	95.7	**4.9**
ResNet-18	CIFAR-100, $90 \to 70$ / $90 \to 80$	93.0 / 94.0	**0.0 / 0.0**
ResNet-50	ImageNet, 132-pair mean	$\sim$98	$\sim$8

V1.7). Unprotected ResNet-18 is at random by an eight percent BER while the protected model holds near 95% until the code limit near twenty percent (Figure 4, Appendix H). Proposition 1 predicts this: with parity in clean carriers a codeword exposes 48 bits, so the failure probability is 1.3×10^{-4} at 3%, 3.4×10^{-6} at 6%, and 0.15 at 20%, where the rescue curve falls.

The same pipeline applies unchanged to ImageNet with $K = 4$ at a 40% budget (Table 1, block 1): ResNet-50, MaxViT-T, RegNet-Y-16GF, Wide ResNet-50-2, and Swin-T hold a normalized accuracy of 0.96 or above at twelve percent, and the plain ViT-B/16 needs a five-plane budget and holds 0.97 through ten percent and 0.91 at twelve.

5.2 Comparison with State-of-the-Art Embedded ECC

Table 1 (block 2; full table in Table 14, Appendix H) compares SANAI with five prior embedded ECC schemes under their burst-error protocol: weight nulling and opportunistic parity collapse by six percent (normalized retention 0.01 and 0.06), value-aware parity and E3-CODE retain 0.58 and 0.72, and E3-CODE-H, the strongest, 0.94. Because the reported baselines use lower-accuracy reference models (82.0% clean for ResNet-18 on CIFAR-10 versus our 95%), the comparable metric is normalized retention: SANAI reaches 0.99 at 3% and 0.97 at 6%, above E3-CODE-H, so the advantage is retention, not a higher start. With $K = 4$, MobileNet-V2 and EfficientNet-B0 reach 0.95 and 0.97 at six percent. On the critical-BER metric, the highest rate at which accuracy stays

near clean, SANAI reaches 6×10^{-2} on ResNet-50/ImageNet, $300{,}000\times$ the unprotected 2×10^{-7} and $4\times$ beyond E3-CODE-H at 1.5×10^{-2}.

Access control: parity is unreadable without the key. We protect a model once and attempt recovery with the correct host key and a wrong key across a 0 to 20% sweep (rows V2.14 and V2.15 of Master Table 5; Figure 4, Appendix H). With the correct key ResNet-18 on CIFAR-10 stays above 94% to ten percent; with the wrong key the de-projected parity is noise and accuracy collapses to 44.6% by three percent and to chance by eight. On ImageNet the wrong key yields essentially zero from the first error level, so a copied model cannot be restored under faults and the protection doubles as a lightweight anti-piracy control.

5.3 PROGRESSIVE AND TARGETED BIT-FLIP ATTACKS ON VISION MODELS

PBS concentrates on the most sensitive weights, which forces it into the protected set: 76 to 100% of its flips land on SANAI-protected weights (Table 1, block 3a). Under the adjacent layout the same 500 flips overload the code: 9 to 49 codewords exceed $t = 4$ across six ResNet settings, peaking at 10 to 15 faults in one, and accuracy collapses to 0 to 35%; more coverage does not help, so the problem is grouping, not budget. The compact networks are harder: at $t = 4$ MobileNet-V2 reaches only 17%, and even the strong code leaves them at 64 and 10% on CIFAR-10 (Master Table 6).

Keyed interleaving removes the concentration itself (Table 1, block 3b; Figure 7): with the identical flipped bits, uncorrectable codewords fall to zero on five of six settings and the maximum faults per codeword to 2 or 3, within $t = 4$, as Theorem 1 predicts, and accuracy returns to the clean range. The exception is ResNet-18 on ImageNet: every codeword is corrected but accuracy stays at 1.5% because at 10% coverage the unprotected 90% of the model is itself corrupted by 500 flips, a coverage limit rather than a code failure (10 to 500 flip sweep in Table 15, Appendix I). T-BFA (Table 1, block 3c) must edit the classifier weights, where 97% of its flips land and which SANAI protects: ASR collapses from 93 to 98% to 0 to 8% (Figure 11, Appendix I).

5.4 LARGE LANGUAGE MODELS

Random bit errors. Table 2 (blocks 1a and 1b) applies the LLM operating point at 40% coverage. Unprotected models collapse at any nonzero BER (perplexity at the 10^6 ceiling below 3B, 10^2 to 10^5 above, accuracy at chance); protected models track clean through six percent (Figure 4, right). Larger models are more robust once protected: Qwen2.5-7B, Llama-3.1-8B, and Llama-2-13B are unchanged to two decimals even at 8%, whereas the two smallest models climb steeply at 8%. Accuracy falls faster than perplexity at 6% on the 7B and 8B because those runs stored part of the parity inside protected layers (the overflow regime of Section 3.6); with clean carriers an 8% injection on Qwen2.5-7B gives perplexity 8.32 against 10^6 unprotected. Llama-3-8B-Instruct holds 83.8% ARC-Easy and Llama-3.2-3B 84.1% at 6% BER.

Optimized sign-bit attacks. AttentionBreaker (Das et al., 2024) uses a gradient-and-magnitude score and an evolutionary search (GenBFA) to find a few weights whose sign bits collapse the model into gibberish; SilentStriker (Xu et al., 2025) keeps text fluent while flipping the answer. We evaluate the published attacks, an *adaptive* variant scored on the SANAI-corrected model, and a *code-aware* attacker that knows G, t, and the layout (Table 2, block 2). In every case SANAI restores the exact clean MMLU and perplexity with bit-identical weights. The adaptive attacker fails by construction: no set of sign flips raises the corrected loss above clean, so flipping all 16,056 population weights on Qwen2-0.5B, or 1.17×10^6 bits on the 8B, leaves the corrected model unchanged; with $G = 12 \leq t$ no sign-bit attack places more than 12 errors in a codeword.

Defeating the code requires $t + 1$ errors in one codeword, which with $G \leq t$ means a second flipped plane on some weight. Master Table 9 prices it on Qwen2-0.5B: under the public adjacent layout, matching the damage of the 26-flip undefended attack needs roughly 600 layout-aware two-plane errors, a $23\times$ increase; with keyed interleaving and no key the errors scatter to at most three per codeword and SANAI restores the clean perplexity at every budget up to 1,300. A key-holding attacker can break codewords again, but each buys less damage (1,170 versus 37,635 at 650 errors).

SilentStriker is stealthy, so accuracy is the damage metric (outputs in Table 16, Appendix J): on Llama-3-8B-Instruct, 160 flips drop MMLU from 70.0 to 37.0%; SANAI flags the tamper and returns

Table 2: LLM results at INT8, strong code, $K = 4$, $G = 12 \leq t$, zero added memory. Block 1: random BER at 40% coverage, mean of three seeds, protected model; unprotected models sit at the 10^6 perplexity ceiling (below 3B) or 10^2 to 10^5 (above) and at chance accuracy at every nonzero BER. The accuracy runs stored part of the parity in protected layers (overflow regime, Section 3.6), so their 6% values are conservative. Block 2: AttentionBreaker with gradient ranking and keyed interleaving, every attack corrected to the exact clean value. Block 3: SilentStriker, attacked / SANAI. Rows trace to Master Tables 7 to 9.

Block 1a. Random BER: protected WikiText-2 perplexity

Model	Params	0%	1%	3%	6%	8%
DistilGPT-2	82M	67.9	69.6	73.9	84.0	1746
GPT-2	124M	42.9	44.0	46.8	61.1	1909
Qwen2-0.5B	0.5B	17.34	17.41	17.41	17.44	18.26
Llama-3.2-1B	1B	13.68	13.83	14.13	14.70	25.32
Qwen2-1.5B	1.5B	11.81	11.83	11.88	11.89	12.35
Llama-3.2-3B	3B	10.36	10.44	10.54	10.74	11.73
Qwen2.5-7B	7B	7.72	7.72	7.72	7.72	7.74
Llama-3.1-8B	8B	7.95	7.95	7.95	7.95	8.04
Llama-2-13B	13B	6.19	6.20	6.19	6.20	6.20

Block 1b. Random BER: protected 5-shot MMLU / ARC-Easy (%)

Model	Params	0%	1%	3%	6%
Qwen2-1.5B	1.5B	58.5 / 85.5	57.8 / 84.5	56.5 / 83.0	54.3 / 80.0
Llama-3.2-3B	3B	62.4 / 87.9	62.2 / 87.1	60.4 / 86.1	58.1 / 84.1
Llama-3-8B-Instruct	8B	68.5 / 92.2	67.7 / 91.6	64.2 / 89.7	57.6 / 83.8
Qwen2.5-7B	7B	77.9 / 96.4	76.6 / 95.4	70.1 / 92.4	51.9 / 71.6

Block 2. AttentionBreaker: MMLU (%) and WikiText-2 perplexity

Model	Attack	Flips	Undef. MMLU	Undef. PPL	SANAI MMLU	SANAI PPL
Qwen2-0.5B	Top-10	10	25.8	1,294	**45.6**	**17.65**
	GenBFA	26	23.8	11,935	**45.6**	**17.65**
	Adaptive	16,056	27.0	$>10^6$	**45.6**	**17.65**
Llama-3.2-1B	GenBFA	1,308	28.4	542	**31.6**	**14.20**
	Adaptive	20,971	28.4	18,835	**31.6**	**14.20**
Qwen2.5-7B	Top-500	500	25.0	31,047	**79.2**	**7.74**
Llama-3-8B-Instruct	Top-10	10	27.0	64,978	**71.0**	**10.89**
	GenBFA	1,018,082	24.4	504	**71.0**	**10.89**
	Adaptive	1,174,405	23.6	916	**71.0**	**10.89**

Block 3. SilentStriker (stealthy): attacked / SANAI

Model	Params	MMLU	ARC	HellaSwag	TruthfulQA	PPL
Qwen2-0.5B	0.5B	43.0 / **45.0**	27.0 / **28.5**	38.5 / **45.0**	23.0 / **22.0**	20.98 / **17.65**
Llama-3.2-1B	1B	22.0 / **31.0**	30.5 / **31.5**	49.0 / **52.5**	25.0 / **24.0**	21.02 / **14.20**
Qwen2.5-7B	7B	76.0 / **78.5**	40.5 / **42.0**	63.0 / **63.0**	32.5 / **29.5**	8.05 / **7.74**
Llama-3-8B-Instruct	8B	37.0 / **70.0**	32.5 / **42.5**	48.0 / **59.0**	27.5 / **30.5**	12.3 / **10.89**

every metric to clean on all four models (Table 2, block 3). Over 1 to 100 flips on Qwen2-0.5B, every attack is detected and corrected bit-exactly in 20 of 20 trials, with no false alarm. Two ablations matter (Master Table 8): a weight-magnitude ranking never selects the small attention projections the attacks target, so the model collapses as if undefended, whereas the gradient-informed probe selects them from 10% coverage; and $K = 3$ ($G = 16 > t$) reopens sign-only overloads.

5.5 FINE-TUNING ROBUSTNESS AND ABLATIONS

Fine-tuning. Fine-tuning ResNet-18 and ResNet-50 on CIFAR-10 for ten epochs leaves the parity bit-exact at 10^{-4} and 10^{-3}; only at 10^{-2} does the earliest convolution drift past $\Delta/2$ (0.003 to 0.004% of parity bits), and the surviving parity still corrects a 12% BER to 93.8% and 94.7% (Figure 13, Appendix K).

Ablations. Appendix L (Table 19) isolates each choice.. *Coverage:* 19.3% to 52.8% buys 0.5 points at 6% and 2 at 12% for 2.7× the parity (rows V1.1, V1.24). *Code:* on ResNet-50/ImageNet, $t = 4$ ($f = 3.1\%$) collapses by 6% BER, $t = 8$ (6.3%) falls to 9.2% at 12%, $t = 12$ (9.4%) holds 77.3%, and $m = 8$, $t = 20$ (7.8%) trails $t = 12$ despite the larger t (Remark 1; rows V1.26

to V1.29). *Planes:* $K = 4$ lifts ResNet-50 on ImageNet from 76.5 to 80.9% at 12% BER while leaving ResNet-18 unchanged (rows V1.16 to V1.19). *Replication:* $R \in \{1, 2, 3\}$ at capacity-limited coverage leaves protected accuracy unchanged (94.84, 94.80, 94.78 at 12%), so $R = 1$ is the default (rows V1.30 to V1.32). *Precision:* INT6 ResNet-18/CIFAR-10 holds 93.6% at 12% (row V1.33); INT4 sits past the crossing of Eq. 6.

6 Hardware Realization

We implement two streaming checkers on an 8×8 output-stationary systolic array, synthesized with Synopsys Design Compiler on gscl45nm (Figure 14); both work on 16-weight blocks in fetch order, whiten the parity with a fuse-seeded keystream, and tap the weight path read-only, flagging a nonzero syndrome. The checker is model-agnostic: its area, timing, and energy per block do not depend on the network, and per-inference cost scales with the weight count; the evaluated workload, an INT8 CIFAR-10 CNN, fits the on-chip ROM and permits a bit-exact full-inference check. Both work on blocks of 16 consecutive weights in fetch order, whiten the parity with a fuse-seeded keystream, and tap the weight path read-only, raising a flag when a syndrome is nonzero.

Proposition 3 (Hardware reduction of STDM)**.** *Let $L_s = 1$, one bit per symbol, lattice step Δ equal to one quantization step, and dither $d = \frac{\Delta}{2} z$ with $z \in \{0, 1\}$ a keystream bit. Then the embedding of Eq. 4 writes $\beta \oplus z$ into the least significant bit of the carrier weight, the reader returns that bit XORed with z, and the perturbation is $|\delta w| \leq \Delta/2$ exactly as in the general case.*

Design A is this proposition in silicon: each block is one BCH$(126, 112, t{=}2)$ codeword over $\mathrm{GF}(2^7)$ whose payload is the top seven bits of each weight and whose 14 parity bits occupy bit 0 of 14 weights, so any single flip is a weight-one codeword error and $d_{\min} = 5$ detects it at every bit position, including the sign bit. *Design B* computes the STDM projection of the block on an integer lattice; its detection is magnitude dependent (100% at bit 6, 50% at bit 5, 0% at the sign bit, which moves the projection by exactly two lattice periods), which is why Design A is primary. Both coexist with the software code in the same weights and both detect and flag rather than correct: an in-line correcting variant needs a 140 KB shadow buffer, the whole weight file, and silently miscorrects 44 to 47% of blocks above capacity, so the flag hands recovery to the software decoder. Design A costs 0.41% area, 0.63% dynamic power, zero stall cycles, and zero memory, and closes at 1.27 ns (Table 21, Appendix M). Against a conventional per-block ECC checker synthesized under the same constraints (same code, parity from a separate SRAM port), Design A costs 13% more logic than the detect-only variant but no parity array (1.22 MB for ResNet-18), closes at 1.27 ns against 2.51 ns, and is 40% smaller and $2.6\times$ faster than the correcting variant (Table 22); the protected model is byte-identical to the original instead of 10.94% larger and fetches 9.86% fewer bits per weight, a modeled 195 μJ per ResNet-18 inference at DRAM energies (Han et al., 2016).

7 Discussion, Limitations, and Conclusion

The curves of Section 5.1 use a localized model in which carriers stay clean, as for voltage scaling and RowHammer; under a uniform model that corrupts the parity-carrying layers too, keyed parity placement keeps ResNet-18/CIFAR-10 at 89.3% at 12% BER, against 78.8% with sequential parity and 10.8% unprotected (Table 18, Appendix L). Proposition 2 guards against an adversary who holds the weights but not the key; the honest limit of any ECC is the code-aware multi-plane attacker (Section 5.4).

In summary, SANAI provides robust protection for sensitive model-weight bits with zero additional memory overhead. It achieves this by embedding parity bits directly into model weights and distributing them across a two-dimensional space using a scalable STDM scheme. SANAI is particularly effective in high-BER scenarios, where alternative embedding schemes either fail or incur substantial accuracy degradation. It maintains near-clean model accuracy at BERs of up to 12%, corrects progressive, targeted, and sign-bit attacks, and reduces the hardware overhead of the checker to only 0.41% in area. To the best of our knowledge, SANAI is the first **secure ECC** approach designed to defend against bit-flip attacks targeting critical model-weight bits with the goal of bypassing ECC protection. Furthermore, SANAI is scalable across diverse model architectures and sizes and incurs minimal hardware footprint.

AI USE STATEMENT

In this work, we used generative AI tools for editing and condensing the manuscript text and for checking LaTeX formatting. We have not used generative AI tools for generating research ideas, designing experiments, producing results, or writing experimental code; the remaining required disclosure tasks are not applicable to this work. All AI-assisted text was reviewed and verified by the authors, who take responsibility for the final content of this work, including text, claims, and artifacts.

ETHICS STATEMENT

This work studies defenses against memory faults and bit-flip attacks on deployed neural networks. The attack implementations we use are re-implementations of published, code-available attacks and are evaluated only on public models and benchmarks; we release no new attack capability. The key-gated recovery mechanism is presented as fault tolerance with lightweight access control and is not claimed to be a cryptographic guarantee (Section 7).

REPRODUCIBILITY STATEMENT

Algorithm 1 (Appendix B) provides the full protect and restore procedure, Appendix C gives proofs of all stated results, Appendix E lists models, datasets, quantization, and every operating point, and Appendix A presents four master tables in which every number in the main text is a tagged row with its full configuration. Complete source code of SANAI, protected checkpoints, the fault-injection harness, the attack re-implementations, the checker RTL (hardware implementation) with its testbench and synthesis scripts are provided as anonymized supplementary material; every table is regenerated by scripts from the checkpoints and the seeds recorded alongside them. The scheme also applies unchanged to exported ONNX graphs (Bai et al., 2019), executing in ONNX Runtime without custom operators (Appendix F).

REFERENCES

Yossi Adi, Carsten Baum, Moustapha Cisse, Benny Pinkas, and Joseph Keshet. Turning your weakness into a strength: Watermarking deep neural networks by backdooring. In *27th USENIX Security Symposium*, pp. 1615–1631, 2018.

Junjie Bai, Fang Lu, Ke Zhang, et al. ONNX: Open neural network exchange. `https://github.com/onnx/onnx`, 2019.

Robert C Baumann. Radiation-induced soft errors in advanced semiconductor technologies. *IEEE Transactions on Device and Materials Reliability*, 5(3):305–316, 2005.

Julia Borghoff, Anne Canteaut, Tim Güneysu, Elif Bilge Kavun, Miroslav Knezevic, Lars R Knudsen, Gregor Leander, Ventzislav Nikov, Christof Paar, Christian Rechberger, et al. PRINCE: A low-latency block cipher for pervasive computing applications. In *Advances in Cryptology (ASIACRYPT)*, pp. 208–225, 2012.

Raj Chandra Bose and Dwijendra K Ray-Chaudhuri. On a class of error correcting binary group codes. *Information and Control*, 3(1):68–79, 1960.

Stéphane Burel, Adrian Evans, and Lorena Anghel. Zero-overhead protection for CNN weights. In *IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)*, pp. 1–6, 2021.

Nandhini Chandramoorthy, Karthik Swaminathan, Martin Cochet, Arun Paidimarri, Schuyler Eldridge, Rajiv V Joshi, Matthew M Ziegler, Alper Buyuktosunoglu, and Pradip Bose. Resilient low voltage accelerators for high energy efficiency. In *IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pp. 147–158, 2019.

Brian Chen and Gregory W Wornell. Quantization index modulation: A class of provably good methods for digital watermarking and information embedding. *IEEE Transactions on Information Theory*, 47(4):1423–1443, 2001.

Yu-Hsin Chen, Joel Emer, and Vivienne Sze. Eyeriss: A spatial architecture for energy-efficient dataflow for convolutional neural networks. In *Proceedings of the 43rd Annual International Symposium on Computer Architecture (ISCA)*, pp. 367–379, 2016.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? Try ARC, the AI2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.

Bita Darvish Rouhani, Huili Chen, and Farinaz Koushanfar. DeepSigns: An end-to-end watermarking framework for ownership protection of deep neural networks. In *Proceedings of the 24th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, pp. 485–497, 2019.

Sanjay Das, Swastik Bhattacharya, Souvik Kundu, Shamik Kundu, Anand Menon, Arnab Raha, and Kanad Basu. AttentionBreaker: Adaptive evolutionary optimization for unmasking vulnerabilities in LLMs through bit-flip attacks. *arXiv preprint arXiv:2411.13757*, 2024.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 248–255, 2009.

Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. LLM.int8(): 8-bit matrix multiplication for transformers at scale. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

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 (ICLR)*, 2021.

Abhimanyu Dubey et al. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Steven K Esser, Jeffrey L McKinstry, Deepika Bablani, Rathinakumar Appuswamy, and Dharmendra S Modha. Learned step size quantization. In *International Conference on Learning Representations (ICLR)*, 2020.

Song Han, Xingyu Liu, Huizi Mao, Jing Pu, Ardavan Pedram, Mark A Horowitz, and William J Dally. EIE: Efficient inference engine on compressed deep neural network. In *Proceedings of the 43rd Annual International Symposium on Computer Architecture (ISCA)*, pp. 243–254, 2016.

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 (CVPR)*, pp. 770–778, 2016.

Zhezhi He, Adnan Siraj Rakin, Jingtao Li, Chaitali Chakrabarti, and Deliang Fan. Defending and harnessing the bit-flip based adversarial weight attack. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 14095–14103, 2020.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *International Conference on Learning Representations (ICLR)*, 2021.

Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, 1963.

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

Mark Horowitz. Computing's energy problem (and what we can do about it). In *IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*, pp. 10–14, 2014.

Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2704–2713, 2018.

Mojan Javaheripi and Farinaz Koushanfar. HASHTAG: Hash signatures for online detection of fault-injection attacks on deep neural networks. In *IEEE/ACM International Conference On Computer Aided Design (ICCAD)*, pp. 1–9, 2021.

Norman P Jouppi, Cliff Young, Nishant Patil, David Patterson, Gaurav Agrawal, Raminder Bajwa, Sarah Bates, Suresh Bhatia, Nan Boden, Al Borchers, et al. In-datacenter performance analysis of a tensor processing unit. In *Proceedings of the 44th Annual International Symposium on Computer Architecture (ISCA)*, pp. 1–12, 2017.

Venkata Nithin Kamineni, Habibur Rahaman, Ovishake Sen, Baibhab Chatterjee, Swarup Bhunia, and Rickard Ewetz. E3-CODE: Embedded and efficient error-correcting code for error-resilient neural networks. In *Proceedings of the 63rd ACM/IEEE Design Automation Conference (DAC)*, 2026.

Yoongu Kim, Ross Daly, Jeremie Kim, Chris Fallin, Ji Hye Lee, Donghyuk Lee, Chris Wilkerson, Konrad Lai, and Onur Mutlu. Flipping bits in memory without accessing them: An experimental study of DRAM disturbance errors. *ACM SIGARCH Computer Architecture News*, 42(3):361–372, 2014.

Skanda Koppula, Lois Orosa, A Giray Yağlıkçı, Roknoddin Azizi, Taha Shahroodi, Konstantinos Kanellopoulos, and Onur Mutlu. EDEN: Enabling energy-efficient, high-performance deep neural network inference using approximate DRAM. In *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pp. 166–181, 2019.

Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

Seo-Seok Lee and Joon-Sung Yang. Value-aware parity insertion ECC for fault-tolerant deep neural network. In *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 724–729, 2022.

Jingtao Li, Adnan Siraj Rakin, Zhezhi He, Deliang Fan, and Chaitali Chakrabarti. RADAR: Run-time adversarial weight attack detection and accuracy recovery. In *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 790–795, 2021.

Stephanie Lin, Jacob Hilton, and Owain Evans. TruthfulQA: Measuring how models mimic human falsehoods. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 3214–3252, 2022.

Qi Liu, Jieming Yin, Wujie Wen, Chengmo Yang, and Shi Sha. NeuroPots: Realtime proactive defense against bit-flip attacks in neural networks. In *32nd USENIX Security Symposium*, pp. 6347–6364, 2023.

Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 10012–10022, 2021.

Henrique S Malvar and Dinei A F Florêncio. Improved spread spectrum: A new modulation technique for robust watermarking. *IEEE Transactions on Signal Processing*, 51(4):898–905, 2003.

James L Massey. Shift-register synthesis and BCH decoding. *IEEE Transactions on Information Theory*, 15(1):122–127, 1969.

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

ONNX Runtime developers. ONNX Runtime. `https://onnxruntime.ai`, 2018.

Minghai Qin, Chao Sun, and Dejan Vucinic. Robustness of neural networks against storage media errors. *arXiv preprint arXiv:1709.06173*, 2017.

Martin Raab and Angelika Steger. "Balls into bins": A simple and tight analysis. *Randomization and Approximation Techniques in Computer Science (RANDOM)*, pp. 159–170, 1998.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI Technical Report*, 2019.

Ilija Radosavovic, Raj Prateek Kosaraju, Ross Girshick, Kaiming He, and Piotr Dollár. Designing network design spaces. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10428–10436, 2020.

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

Adnan Siraj Rakin, Zhezhi He, and Deliang Fan. TBT: Targeted neural network attack with bit trojan. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 13198–13207, 2020.

Adnan Siraj Rakin, Zhezhi He, Jingtao Li, Fan Yao, Chaitali Chakrabarti, and Deliang Fan. T-BFA: Targeted bit-flip adversarial weight attack. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(11):7928–7939, 2021.

Brandon Reagen, Udit Gupta, Lillian Pentecost, Paul Whatmough, Sae Kyu Lee, Niamh Mulholland, David Brooks, and Gu-Yeon Wei. Ares: A framework for quantifying the resilience of deep neural networks. In *Proceedings of the 55th Annual Design Automation Conference (DAC)*, pp. 1–6, 2018.

Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. MobileNetV2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4510–4520, 2018.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. DistilBERT, a distilled version of BERT: Smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*, 2019.

Bianca Schroeder, Eduardo Pinheiro, and Wolf-Dietrich Weber. DRAM errors in the wild: A large-scale field study. *Communications of the ACM*, 54(2):100–107, 2011.

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

Sergei P Skorobogatov and Ross J Anderson. Optical fault induction attacks. *Cryptographic Hardware and Embedded Systems (CHES)*, pp. 2–12, 2003.

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.

Mingxing Tan and Quoc Le. EfficientNet: Rethinking model scaling for convolutional neural networks. In *International Conference on Machine Learning (ICML)*, pp. 6105–6114, 2019.

Hugo Touvron, Louis Martin, Kevin Stone, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

Zhengzhong Tu, Hossein Talebi, Han Zhang, Feng Yang, Peyman Milanfar, Alan Bovik, and Yinxiao Li. MaxViT: Multi-axis vision transformer. In *European Conference on Computer Vision (ECCV)*, pp. 459–479, 2022.

Yusuke Uchida, Yuki Nagai, Shigeyuki Sakazawa, and Shin'ichi Satoh. Embedding watermarks into deep neural networks. In *Proceedings of the ACM International Conference on Multimedia Retrieval (ICMR)*, pp. 269–277, 2017.

Haotian Xu et al. SilentStriker: Toward stealthy bit-flip attacks on large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

An Yang et al. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*, 2024a.

An Yang et al. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024b.

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*, pp. 1463–1480, 2020.

Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In *British Machine Vision Conference (BMVC)*, 2016.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. HellaSwag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 4791–4800, 2019.

Jeff Zhang, Kartheek Rangineni, Zahra Ghodsi, and Siddharth Garg. ThUnderVolt: Enabling aggressive voltage underscaling and timing error resilience for energy efficient deep learning accelerators. In *Proceedings of the 55th Annual Design Automation Conference (DAC)*, pp. 1–6, 2018.

APPENDIX

A MASTER TABLES

Every number in the main text is a row, or part of a row, of one of the master tables below (V1, V2, A1, L1, L2, L3). Each row is one run with its full configuration, so numbers for the same model that differ across main-text tables can be traced to the knob that differs. Cfg: S = strong code shortened from BCH$(127, 50, t{=}12)$ with 88 parity bits; L = light code shortened from BCH$(127, 99, t{=}4)$ with 32 parity bits. Blank cells were not run. Coverage is the protected weight fraction; R is parity replication; Intl. marks keyed interleaving.

Table 3: Master Table V1: vision runs under random BER with the strong code (INT8 unless noted), top-1 accuracy (%). Rows V1.24 to V1.33 are the ablations of Section 5.5. ImageNet rows use layer-mode selection at a 40% budget; unprotected ImageNet accuracy is below 1% at 12% BER for every model. †five-plane budget.

ID	Model	Data	Quant	Code	K	Cov.	R	0%	1%	3%	6%	8%	10%	12%
V1.1	ResNet-18	C10	PTQ	S	3	19.3%	1	95.09	94.95	94.66	94.54	94.46	94.36	93.06
V1.2	ResNet-18	C10	Jacob	S	3	19.5%	1	95.05	94.99	94.58	94.53	94.41	94.41	93.53
V1.3	ResNet-50	C10	PTQ	S	3	19.2%	1	94.93	94.87	94.79	94.51	94.54	94.10	93.70
V1.4	VGG-16	C10	PTQ	S	3	20.0%	1	90.20	90.12	89.98	89.87	89.72	89.76	89.19
V1.5	ResNet-18	C100	PTQ	S	3	14.0%	1	77.05	77.30	77.10	76.82	76.69	76.11	75.86
V1.6	ResNet-18	C10	LSQ	S	3	18.6%	1	94.98	93.26	90.78	78.60	76.25	63.41	27.38
V1.7	ResNet-18	C10	LSQ	S	3	36.6%	1	94.98	94.86	94.52	93.26	92.73	91.26	85.84
V1.8	ResNet-50	C100	PTQ	S	3	36.1%	1	75.77	75.64	75.36	75.32	75.41	75.36	74.71
V1.9	VGG-16	C100	PTQ	S	3	23.7%	1	65.92	65.91	65.74	65.72	65.74	65.56	64.69
V1.10	EfficientNet-B0	C100	PTQ	S	4	30.0%	1	62.86	63.29	62.49	61.96	61.46	59.97	53.22
V1.11	MaxViT-T	IN	PTQ	S	4	40.9%	1	88.14	88.14	88.06	88.05	87.97	87.69	87.55
V1.12	RegNet-Y-16GF	IN	PTQ	S	4	≈40%	1	85.78	85.66	85.50	85.05	84.64	84.62	84.48
V1.13	Wide ResNet-50-2	IN	PTQ	S	4	≈40%	1	84.00	83.91	83.97	84.06	83.91	83.58	83.58
V1.14	Swin-T	IN	PTQ	S	4	≈40%	1	86.61	86.50	86.28	85.42	85.34	84.92	83.44
V1.15	ViT-B/16	IN	PTQ	S	5	43.5%	1	85.59	85.50	85.42	85.28	84.44	83.25	77.89
V1.16	ResNet-18	IN	PTQ	S	3	40.4%	1	76.14	76.05	76.14	75.95	75.70	75.78	75.70
V1.17	ResNet-18	IN	PTQ	S	4	40.4%	1	76.14	76.20	76.12	76.16	75.95	75.98	75.91
V1.18	ResNet-50	IN	PTQ	S	3	40.9%	1	82.83	81.83	81.03	79.58	78.58	79.05	76.45
V1.19	ResNet-50	IN	PTQ	S	4	40.9%	1	82.83	82.72	82.52	82.41	82.06	81.31	80.88
V1.24	ResNet-18	C10	PTQ	S	3	52.8%	1	95.06	95.00	94.91	95.02	94.84	94.95	95.02
V1.25	ResNet-18	C10	PTQ	S	4	52.8%	1	95.06	95.02	95.00	95.05	94.94	94.95	94.89
V1.26	ResNet-50	IN	PTQ	$t{=}4$	3	23.4%	1	82.83	81.53	64.30	2.61	0.34	0.00	0.05
V1.27	ResNet-50	IN	PTQ	$t{=}8$	3	23.4%	1	82.83	82.31	80.95	78.41	65.55	33.28	9.17
V1.28	ResNet-50	IN	PTQ	$t{=}12$	3	23.4%	1	82.83	82.09	81.03	80.73	80.09	78.86	77.30
V1.29	ResNet-50	IN	PTQ	$t{=}20, m{=}8$	3	23.4%	1	82.83	82.34	81.33	80.88	79.84	78.11	75.67
V1.30	ResNet-50	C10	PTQ	S	3	39.8%	1	94.86	94.88	94.69	94.97	94.67	94.75	94.84
V1.31	ResNet-50	C10	PTQ	S	3	20.4%	2	94.86	94.91	94.86	94.86	94.89	94.83	94.80
V1.32	ResNet-50	C10	PTQ	S	3	15.1%	3	94.86	94.92	94.88	94.95	94.84	94.97	94.78
V1.33	ResNet-18	C10 (INT6)	PTQ	S	4	52.8%	1	95.08	95.00	94.86	94.73	94.56	94.20	93.64

Table 4: Master Table V1b: ResNet-18 / CIFAR-10 under weight-level sensitivity bins (block A; high = top 1% of weights by attack-damage proxy, mid = next 8%; V1.35 had host bits exposed and 11 overflow carriers) and under the uniform fault model that corrupts the carrier layers at the same BER (block B; strong code, $K = 3$, 19% coverage). Protected (unprotected) top-1 (%).

Block A. Weight-level sensitivity bins

ID	Bins protected	Layers	K	Cov.	3%	6%	T-BFA ASR unprot. / prot.
V1.34	high only	14	4	24.8%	94.95 (69.16)	95.01 (15.03)	95.3 / 4.5
V1.35	high + half mid	16	4	56.5%	93.32 (66.92)	79.11 (17.06)	95.3 / 8.4

Block B. Uniform fault model: carrier layers corrupted at the same BER

ID	Parity placement	K	Cov.	0%	6%	12%
V1.36	interleaved	3	19%	95.06	93.92 (10.25)	**89.34** (10.78)
V1.37	sequential	3	19%	95.06	93.83 (10.25)	78.83 (10.78)

B ALGORITHM AND EMBEDDING DETAILS

Algorithm 1 states SANAI end to end. Figure 2 illustrates the two-scale lattice used when B parity bits are embedded per symbol: a group of B bits is read as a level in $[0, 2^B - 1]$, the coarse step $\Delta 2^B$ selects the cell, the fine step Δ places one sub-level per value inside each cell, and decoding rounds to the nearest fine sub-level, $\hat{\beta} = \mathrm{round}((p - d)/\Delta) \bmod 2^B$. Setting $L_s = 1$ recovers plain QIM capacity of one bit per carrier weight while keeping key dependence; raising L_s averages L_s independent low-bit perturbations, trading capacity for robustness of the hidden parity.

Table 5: Master Table V2: vision runs under the burst protocol of Kamineni et al. (2026) with the strong code (block A), under random BER with the light code at the attack operating point (block B), and under key-gated recovery with $L_s = 2$ (block C). Top-1 (%); unprotected in parentheses where measured.

Block A. Burst protocol, strong code

ID	Model	Data	K	Cov.	0%	1.5%	3%	6%
V2.1	ResNet-18	C10	3	20%	95.09	94.9	94.7	94.5
V2.2	ResNet-50	C10	3	20%	94.93	94.9	94.8	94.5
V2.3	MobileNet-V2	C10	4	20%	79.27	78.2	76.6	75.3
V2.4	EfficientNet-B0	C10	4	20%	84.23	83.8	83.0	81.4
V2.5	ResNet-18	IN	3	20%	76.14	75.5	75.1	73.3
V2.6	ResNet-50	IN	3	20%	82.83	82.2	81.6	80.5
V2.7	MobileNet-V2	IN	4	20%	78.39	78.4	78.1	77.8
V2.8	EfficientNet-B0	IN	4	20%	83.42	82.5	80.6	76.2

Extended to 12% burst BER on five settings: V2.1 93.1, V2.2 93.7, V2.3 67.1, V2.4 75.0, V2.6 62.6.

Block B. Random BER, light code, keyed interleaving; protected (unprotected)

ID	Model	Data	K	Cov.	0%	1%	3%	6%	12%
V2.9	ResNet-18	C10	4	10%	95.1	95.1 (87.5)	94.9 (52.5)	94.8 (20.4)	94.7 (10.0)
V2.10	ResNet-18	IN	4	10%	76.1	75.5 (0.2)	73.4 (0.0)	72.4 (0.0)	46.2 (0.0)
V2.11	ResNet-50	IN	4	10%	82.8	82.6 (0.5)	81.8 (0.8)	79.9 (0.0)	78.1 (1.2)
V2.12	MobileNet-V2	IN	4	10%	78.4	77.2 (0.0)	72.6 (0.3)	64.1 (0.0)	6.3 (0.1)
V2.13	EfficientNet-B0	IN	4	10%	83.4	82.1 (0.0)	79.7 (0.0)	73.3 (0.1)	37.1 (0.3)

Block C. Key-gated recovery, strong code, $K = 3$

ID	Model	Data	Key	Cov.	0%	1.5%	3%	6%	12% / 20%
V2.14	ResNet-18	C10	correct	19.3%	95.09	94.89	94.66	94.54	93.06 / 44.10
V2.15	ResNet-18	C10	wrong	19.3%	94.88	77.43	44.58	22.91	10.90 / 9.98
V2.16	ResNet-18	IN	correct	20%	76.14	75.50	75.08	73.28	59.42 / 0.02
V2.17	ResNet-18	IN	wrong	20%	67.30	0.00	0.02	0.00	0.62 / 0.02

Table 6: Master Table A1: vision runs under adversarial bit-flip attacks (INT8). ResNets use the light code, $K = 4$, 10% coverage; compact models use the strong code with weight-level selection. Top-1 (%) for PBS; cw$> t$ is the number of uncorrectable codewords and max/cw the largest fault count in one; ASR is the T-BFA success rate (%). Rows A1.15 (100 flips) and A1.16 are from an earlier run; the same run at 19% coverage gave 65.7%.

Block A. Progressive bit search, 500 flips, keyed interleaving; ResNets with the light code in layer-mode selection, compact networks with the strong code in weight-mode selection

ID	Model	Data	Code	Cov.	Unprot. acc.	Prot. acc.	cw$> t$	max/cw
A1.1	ResNet-18	C10	light t=4	10%	10.06	94.9	0	3
A1.2	ResNet-50	C10	light t=4	10%	35.00	94.8	0	3
A1.3	MobileNet-V2	C10	strong t=12	10%	9.94	64.1	1†	5†
A1.4	EfficientNet-B0	C10	strong t=12	10%	10.00	10.3	0†	3†
A1.5	ResNet-18	C100	light t=4	10%	1.16	75.5	0	2
A1.6	ResNet-50	C100	light t=4	10%	0.94	75.8	0	2
A1.7	MobileNet-V2	C100	strong t=12	10%	1.02	35.0	0†	2†
A1.8	EfficientNet-B0	C100	strong t=12	10%	0.98	60.9	3†	8†

Block B. Codeword layout under PBS, 500 flips: adjacent versus keyed interleaving (same flipped bits)

ID	Model	Data	Flips	Cov.	Prot. acc. adj / keyed	cw$> t$ adj / keyed	max/cw adj / keyed
A1.9	ResNet-18	C10	500	10%	10.06 / 94.86	22 / 0	14 / 3
A1.10	ResNet-50	C10	500	10%	35.00 / 94.81	14 / 0	15 / 3
A1.11	ResNet-18	C100	500	10%	1.16 / 75.48	26 / 0	13 / 2
A1.12	ResNet-50	C100	500	10%	0.94 / 75.77	49 / 0	10 / 2
A1.13	ResNet-18	IN	500	10%	0.00 / 1.53	14 / 0	12 / 2
A1.14	ResNet-50	IN	500	10%	1.70 / 81.88	9 / 5	12 / 2
A1.15	ResNet-18	C10	100	10%	66.0 / 95.1	3 / 0	12 / 2

Block D. Fault-map pass on the compact networks, 500 flips, light code $t = 4$, the fault-map tool's own weight selection

ID	Model	Data	Flips	Layout	cw$> t$	max/cw
A1.21	MobileNet-V2	C10	500	adjacent / keyed	42 / 1	10 / 5
A1.22	MobileNet-V2	C100	500	adjacent / keyed	35 / 0	10 / 2
A1.23	EfficientNet-B0	C10	500	adjacent / keyed	17 / 0	14 / 3
A1.24	EfficientNet-B0	C100	500	adjacent / keyed	23 / 3	24 / 8

Block C. Targeted attack (T-BFA), strong code, high-bin weight selection

ID	Model	Data	Pair	Cov.	Flips	ASR unprot.	ASR prot.
A1.17	ResNet-18	C10	$3 \to 5$	24.8%	15	95.3 / 95.7	4.5 / 4.9
A1.18	ResNet-18	C100	$90 \to 70$	10%	10 to 20	93.0	0.0
A1.19	ResNet-18	C100	$90 \to 80$	10%	10 to 20	94.0	0.0
A1.20	ResNet-50	IN	132-pair mean	10%	10 to 20	∼98	∼8

Complexity. Let C_{eval} be the cost of one validation pass. The protect phase costs $O(L\, C_{\text{eval}})$ for probing, $O(L \log L)$ for selection, $O(N_{\text{cw}} nt)$ for BCH encoding, and $O(W_c)$ for STDM embedding. The restore phase costs $O(W_c)$ for de-projection, $O(N_{\text{cw}}(nt + t^2))$ for Berlekamp-Massey decoding (Massey, 1969), and $O(W_p K)$ for splicing, linear in model size and independent of the fault rate. The mixed-integer programs of prior embedded ECC are NP-hard in general and need commercial solvers whose runtime grows sharply with layer count.

Table 7: Master Table L1: LLM runs under random BER (INT8, strong code, $K = 4$, 40% coverage, magnitude ranking, adjacent grouping, three seeds). P = protected, U = unprotected (perplexity capped at 10^6); perplexity rows give mean $\pm$ std, accuracy rows are 5-shot on 2,000 questions to 6% BER. Perplexity runs keep carriers clean; accuracy runs let parity overflow into protected layers (Section 3.6), so the 7B and 8B accuracy at 6% is conservative.

ID	Model	Metric	P/U	0%	1%	3%	6%	8%
L1.1	DistilGPT-2	PPL W2	P	67.86	69.59 $\pm$ 0.85	73.93 $\pm$ 1.22	84.03 $\pm$ 2.27	1746 $\pm$ 1346
L1.2		PPL W103	P	61.64	63.16 $\pm$ 0.68	67.39 $\pm$ 1.23	76.77 $\pm$ 2.96	1721 $\pm$ 1358
L1.3		PPL W2	U	67.86	10^6	10^6	10^6	10^6
L1.4	GPT-2	PPL W2	P	42.91	44.00 $\pm$ 0.32	46.77 $\pm$ 0.37	61.07 $\pm$ 4.87	1909 $\pm$ 667
L1.5		PPL W103	P	38.43	39.34 $\pm$ 0.04	42.26 $\pm$ 0.30	52.99 $\pm$ 2.91	1695 $\pm$ 407
L1.6		PPL W2	U	42.91	10^6	10^6	10^6	10^6
L1.7	Qwen2-0.5B	PPL W2	P	17.34	17.41 $\pm$ 0.03	17.41 $\pm$ 0.04	17.44 $\pm$ 0.03	18.26 $\pm$ 0.25
L1.8		PPL W103	P	15.22	15.27 $\pm$ 0.02	15.29 $\pm$ 0.03	15.33 $\pm$ 0.03	15.94 $\pm$ 0.15
L1.9		PPL W2	U	17.34	10^6	10^6	10^6	10^6
L1.10	Qwen2-1.5B	PPL W2	P	11.81	11.83 $\pm$ 0.00	11.88 $\pm$ 0.01	11.89 $\pm$ 0.02	12.35 $\pm$ 0.18
L1.11		PPL W103	P	10.92	10.94 $\pm$ 0.00	10.97 $\pm$ 0.00	10.97 $\pm$ 0.02	11.35 $\pm$ 0.15
L1.12		PPL W2	U	11.81	10^6	10^6	10^6	10^6
L1.13		MMLU / ARC-E	P	58.5 / 85.5	57.8 $\pm$ 0.3 / 84.5 $\pm$ 0.19	56.5 $\pm$ 0.4 / 83.0 $\pm$ 0.37	54.3 $\pm$ 0.5 / 80.0 $\pm$ 0.41	(to 6%)
L1.14	Llama-3.2-1B	PPL W2	P	13.68	13.83 $\pm$ 0.04	14.13 $\pm$ 0.08	14.70 $\pm$ 0.30	25.32 $\pm$ 9.44
L1.15		PPL W103	P	12.66	12.77 $\pm$ 0.04	13.06 $\pm$ 0.05	13.54 $\pm$ 0.21	23.98 $\pm$ 9.58
L1.16		PPL W2	U	13.68	9.5×10^5	10^6	10^6	10^6
L1.17	Llama-3.2-3B	PPL W2	P	10.36	10.44 $\pm$ 0.01	10.54 $\pm$ 0.02	10.74 $\pm$ 0.03	11.73 $\pm$ 0.14
L1.18		PPL W103	P	9.73	9.78 $\pm$ 0.01	9.88 $\pm$ 0.02	10.03 $\pm$ 0.05	10.94 $\pm$ 0.17
L1.19		PPL W2	U	10.36	2.2×10^5	5.4×10^5	5.8×10^5	5.8×10^5
L1.20		MMLU / ARC-E	P	62.4 / 87.9	62.2 $\pm$ 0.3 / 87.1 $\pm$ 0.29	60.4 $\pm$ 0.1 / 86.1 $\pm$ 0.54	58.1 $\pm$ 0.7 / 84.1 $\pm$ 0.16	(to 6%)
L1.21	Qwen2.5-7B	PPL W2	P	7.72	7.72 $\pm$ 0.00	7.72 $\pm$ 0.00	7.72 $\pm$ 0.00	7.74 $\pm$ 0.02
L1.22		PPL W2	U	7.72	3.9×10^3	2.7×10^4	6.6×10^4	4.1×10^5
L1.23		MMLU / ARC-E	P	77.9 / 96.4	76.6 $\pm$ 0.4 / 95.4 $\pm$ 0.14	70.1 $\pm$ 0.5 / 92.4 $\pm$ 0.60	51.9 $\pm$ 0.9 / 71.6 $\pm$ 1.51	(to 6%)
L1.24	Llama-3-8B-Instruct	MMLU / ARC-E	P	68.5 / 92.2	67.7 $\pm$ 0.4 / 91.6 $\pm$ 0.04	64.2 $\pm$ 0.5 / 89.7 $\pm$ 0.29	57.6 $\pm$ 0.8 / 83.8 $\pm$ 0.40	(to 6%)
L1.25	Llama-3.1-8B	PPL W2	P	7.95	7.95 $\pm$ 0.00	7.95 $\pm$ 0.00	7.95 $\pm$ 0.00	8.04 $\pm$ 0.11
L1.26		PPL W2	U	7.95	3.9×10^2	8.2×10^4	3.1×10^4	6.7×10^4
L1.27	Llama-2-13B	PPL W2	P	6.19	6.20 $\pm$ 0.00	6.19 $\pm$ 0.00	6.20 $\pm$ 0.00	6.20 $\pm$ 0.01
L1.28		PPL W2	U	6.19	3.8×10^2	3.0×10^3	6.3×10^3	7.8×10^3

Unprotected MMLU / ARC-E is 24 to 26% (four-way chance) on every model at every nonzero BER.

Table 8: Master Table L2: LLM runs under AttentionBreaker (INT8, strong code, $K = 4$, $G = 12$). Rank: grad = gradient-weighted magnitude, mag = weight magnitude; Intl.: adj = adjacent codewords, key = keyed interleaving; "In" = fraction of flipped bits inside the protected set. The largest error count in one codeword, where logged, never exceeded $t = 12$ (Qwen2-0.5B rows 1 to 12; Llama-3.2-1B adaptive 5; Llama-3-8B GenBFA and adaptive ≤ 12). Row L2.8 is the ranking ablation; the last block is the coverage ablation.

ID	Model	Attack	Flips	Rank	Cov.	Intl.	U MMLU	U PPL	SANAI MMLU	SANAI PPL	In
L2.1	Qwen2-0.5B	Top-3	3	grad	40%	key	45.2	20.9	45.6	17.65	3/3
L2.2		Top-10	10	grad	40%	key	25.8	1,294	45.6	17.65	10/10
L2.3		Top-28	28	grad	40%	key	26.2	4,429	45.6	17.65	28/28
L2.4		Top-100	100	grad	40%	key	26.2	964,460	45.6	17.65	100/100
L2.5		Top-500	500	grad	40%	key	27.0	$> 10^6$	45.6	17.65	500/500
L2.6		GenBFA	26	grad	40%	key	23.8	11,935	45.6	17.65	26/26
L2.7		Adaptive	16,056	grad	40%	key	27.0	$> 10^6$	45.6	17.65	all
L2.8		Top-10	10	mag	40%	key	25.8	1,294	25.8	1,294	0/10
L2.9		Top-10	10	grad	10%	adj	25.8	1,294	45.6	17.65	10/10
L2.10	Llama-3.2-1B	Top-10	10	grad	40%	key	28.6	16.2	31.6	14.20	all
L2.11		Top-100	100	grad	40%	key	25.0	143	31.6	14.20	all
L2.12		GenBFA	1,308	grad	40%	key	28.4	542	31.6	14.20	all
L2.13		Adaptive	20,971	grad	40%	key	28.4	18,835	31.6	14.20	all
L2.14	Qwen2.5-7B	Top-3	3	grad	40%	key	23.5	7.84	79.2	7.74	all
L2.15		Top-28	28	grad	40%	key	23.5	12.97	79.2	7.74	all
L2.16		Top-100	100	grad	40%	key	25.5	1,497	79.2	7.74	all
L2.17		Top-500	500	grad	40%	key	25.0	31,047	79.2	7.74	all
L2.18	Llama-3-8B-Inst.	Top-3	3	grad	40%	key	63.8	17.0	71.0	10.89	all
L2.19		Top-10	10	grad	40%	key	27.0	64,978	71.0	10.89	all
L2.20		Top-100	100	grad	40%	key	26.0	3,474	71.0	10.89	all
L2.21		GenBFA	1,018,082	grad	40%	key	24.4	504	71.0	10.89	all
L2.22		Adaptive	1,174,405	grad	40%	key	23.6	916	71.0	10.89	all
L2.26	GPT-2	Top-3 / 10 / 28 / 100	3 to 100	grad	40%	key	PPL only	2,064 / 2,926 / 3,147 / 3,798	PPL only	43.10 at every budget	100/100
L2.27	DistilGPT-2	Top-3 / 10 / 28 / 100	3 to 100	grad	40%	key	PPL only	616 / 2,661 / 8,219 / 5,343	PPL only	67.87 at every budget	100/100

Coverage ablation, Llama-3-8B-Instruct, Top-10 (10 flips), gradient ranking, keyed interleaving; the attack targets `layers.1.mlp.down_proj`

ID	Model	Cov.	Attacked layer in protected set	Undef. MMLU / PPL	SANAI MMLU	SANAI PPL	In	Recovered
L2.23	Llama-3-8B-Inst.	10%	no	27.0 / 64,978	27.0	64,978	0/10	no
L2.24		20%	no	27.0 / 64,978	27.0	64,978	0/10	no
L2.25		30%	yes	27.0 / 64,978	**71.0**	**10.89**	10/10	yes, exact

C PROOFS

Proposition 1. The decoder corrects any pattern of at most t errors and fails on any pattern of more, so the failure event is $\{X > t\}$ with $X \sim \mathrm{Bin}(n, p)$, which gives the binomial tail. Writing X/n as the mean of n independent Bernoulli(p) variables and applying Hoeffding's inequality (Hoeffding, 1963) to the deviation $f - p > 0$ gives $\Pr[X/n > f] \leq \exp(-2n(f-p)^2)$. □

Lemma (random-fault correctability). Let each codeword hold GK protected bits flipped independently with probability p. Then $f_c \sim \mathrm{Bin}(GK, p)$ and $\Pr[f_c > t] = \sum_{i>t} \binom{GK}{i} p^i (1-p)^{GK-i}$, which for $GKp \ll t$ is $O((GKp)^{t+1})$. With $GK = 96$, $t = 4$, $p = 0.06$ the per-codeword failure probability is small and the model-level accuracy is preserved, matching the flat protected curves in Master Table 3.

Table 9: Master Table L3: LLM runs under the code-aware attacker, SilentStriker, and the detection sweep (INT8, strong code, $K = 4$). Code-aware rows place 13 two-plane errors per targeted codeword and list WikiText-2 perplexity after SANAI at each error budget; blind = interleaved layout unknown to the attacker. SilentStriker rows give attacked / SANAI MMLU and perplexity and the number of the 160 attacked codewords whose syndrome flagged (other metrics in Table 2); every model recovers with zero weights wrong. Detection rows are 20 trials of random sign flips inside the protected set on Qwen2-0.5B at 10% coverage; the clean model raised zero false alarms over 1.19×10^7 codewords.

ID	Model	Attack	Budget	Cov.	Intl.	Result	Detected / exact
L3.1	Qwen2-0.5B	Code-aware, 2-plane	13 / 65 / 130 / 260 / 650 / 1,300	40%	adj	PPL 18.1 / 28.0 / 48.1 / 769 / 37,635 / 93,819	yes / no
L3.2		Code-aware, 2-plane	same budgets	40%	blind	PPL 17.65 at every budget (max/cw ≤ 3)	yes / yes
L3.3		Code-aware, 2-plane	same budgets	40%	key	PPL 18.4 / 34.9 / 50.4 / 137 / 1,170 / 312,778	yes / no
L3.4	Llama-3.2-1B	Code-aware, 2-plane	130 / 260 / 650 / 1,300 / 2,600	40%	adj	PPL 14.9 / 15.8 / 19.1 / 23.0 / n.a.	yes / no
L3.5		Code-aware, 2-plane	same budgets	40%	key	PPL 15.0 / 16.1 / 23.9 / 48.7 / 129.5	yes / no
L3.6	Qwen2-0.5B	SilentStriker	160 flips	60%	key	MMLU 43.0 / 45.0, PPL 20.98 / 17.65, 146 of 160 codewords flagged	yes / yes
L3.7	Llama-3.2-1B	SilentStriker	160 flips	60%	key	MMLU 22.0 / 31.0, PPL 21.02 / 14.20, 157 of 160 codewords flagged	yes / yes
L3.8	Qwen2.5-7B	SilentStriker	160 flips	60%	key	MMLU 76.0 / 78.5, PPL 8.05 / 7.74, 160 of 160 codewords flagged	yes / yes
L3.9	Llama-3-8B-Inst.	SilentStriker	160 flips	60%	key	MMLU 37.0 / 70.0, PPL 12.3 / 10.89	yes / yes
L3.10	Qwen2-0.5B	Random sign flips	1	10%	key	corrected bit-exactly (max/cw 1)	20/20 / 20/20
L3.11		Random sign flips	2	10%	key	corrected bit-exactly	20/20 / 20/20
L3.12		Random sign flips	5	10%	key	corrected bit-exactly	20/20 / 20/20
L3.13		Random sign flips	10	10%	key	corrected bit-exactly	20/20 / 20/20
L3.14		Random sign flips	20	10%	key	corrected bit-exactly	20/20 / 20/20
L3.15		Random sign flips	50	10%	key	corrected bit-exactly	20/20 / 20/20
L3.16		Random sign flips	100	10%	key	corrected bit-exactly	20/20 / 20/20

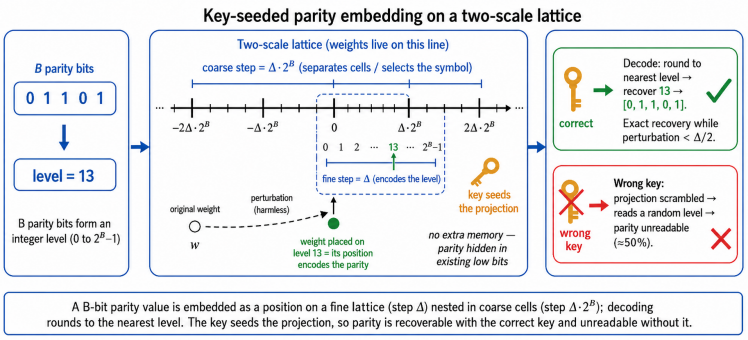


Figure 2: Key-seeded parity embedding on a two-scale lattice. Recovery is exact while the perturbation stays below $\Delta/2$; without the key the parity reads at chance.

Lemma (adversarial concentration). A targeted attacker with a flip budget F maximizes damage by maximizing $\max_c f_c$. Under adjacent grouping all F flips of a sensitive region can be placed in $\lceil F/(GK) \rceil$ codewords, so one codeword may reach $f_c = \min(F, GK) \gg t$. This is the measured behaviour: 500 PBS flips drive 29 codewords above $t = 4$ with a peak of 27 (Table 1).

Proposition 2. (a) With the correct key the reader forms $w' \cdot v + d = p^\star + \delta$ where δ is the read-time perturbation of the projection; $p^\star$ lies on the lattice selected by β and the two lattices are separated by $\Delta/2$, so rounding returns β whenever $|\delta| < \Delta/2$. (b) For an independent pseudo-random $v' \in \{-1, +1\}^{L_s}$, $\mathbb{E}[v \cdot v'] = 0$ and $w' \cdot v'$ is a sum of terms whose signs are independent of β; the parity of its rounded level is a Bernoulli($\frac{1}{2}$) variable independent across groups, so all P_{bits} bits are recovered with probability $2^{-P_{\text{bits}}}$. □

Theorem 1. Since π is uniform and unknown to the attacker, the codeword of each chosen weight is a uniformly random bin among M, independent across flips up to the without-replacement correction, which only tightens the bound. The maximum load of F balls in M bins is $O(F/M + \log M / \log \log M)$ in expectation (Raab & Steger, 1998). For $F \ll M \log M$ the first term vanishes and the second is $O(1)$ for any practical M, so $\max_c f_c \leq t$ with high probability. □

Worked example of the interleaving bound. Take the light code, $t = 4$ and $G = 24$ weights per codeword, and a model of $M = 1000$ codewords. An attacker who knows the code parameters flips $F = 10$ bits in one physical region, intending to overload one codeword. Under adjacent grouping all ten land in that codeword, $f_c = 10 > t$, and it fails; the attacker wins with $t + 1 = 5$ flips. Under a keyed permutation the ten flips fall into the M codewords like balls into bins: $\mathbb{E}[f_c] = F/M = 0.01$ and $\max_c f_c \approx F/M + \ln M / \ln \ln M \approx 3.6 \leq t$, so every codeword is corrected although ten bits were flipped. Setting the bound above t gives $F/M \gtrsim 0.4$, so the blind attacker needs on the order of 400 flips before any codeword fails, against 5 under adjacent grouping. Without the key the attacker cannot direct a flip into a chosen codeword: the probability of landing $t + 1$ blind flips in one specific

Algorithm 1: SANAI: sensitivity-guided bit-plane ECC with keyed interleaving and cross-layer STDM parity hiding

Input: Quantized weights $\{\theta_\ell\}_{\ell=1}^{L}$ at b bits; coverage ρ; planes K; parent code $\mathrm{BCH}(n,k,t)$ over $\mathrm{GF}(2^m)$ with $n = 2^m - 1$, storing p parity bits per k_s-bit message; group size G; lattice step Δ; spread L_s; key κ; probe rate p_0

Output: Protected weights θ' (offline); restored weights $\hat{\theta}$ (read time)

`// Phase 1: protect, offline, run once`

foreach *layer* ℓ **do**
  $S_\ell \leftarrow A_{\text{clean}} - \mathbb{E}_{\epsilon\sim\mathrm{Bern}(p_0)}[A(\theta_\ell \oplus \epsilon)]$ `// Eq. 2`
end
sort layers so that $S_{\ell_{(1)}} \geq \cdots \geq S_{\ell_{(L)}}$; $\mathcal{P} \leftarrow$ largest prefix with $\sum_{\ell\in\mathcal{P}} |\theta_\ell| \leq \rho W$ `// Eq. 3`
$\mathcal{C} \leftarrow$ remaining layers, least sensitive first; $\pi \leftarrow \mathrm{KeyedPerm}(\mathcal{P}, \kappa)$ `// Eq. 5`
foreach *codeword* c *of* G *weights from* $\pi(\mathcal{P})$ **do**
  $M_c \leftarrow \lfloor \theta_c / 2^{b-K} \rfloor$;
  $\pi_c \leftarrow \mathrm{BCH}_{(n,k,t)}(M_c)$
end
$\Pi \leftarrow \pi_{c_1} \| \pi_{c_2} \| \cdots$; **assert** $|\Pi| \leq W_c B / L_s$ `// Eq. 6`
$(v, d) \leftarrow \mathrm{PRNG}(\kappa)$ per carrier group
foreach *parity symbol* $\beta \in \Pi$ *with carrier group* $w \subset \mathcal{C}$ **do**
  $p \leftarrow w \cdot v + d$;
  snap to $p^\star$;
  $w \leftarrow w + \frac{p^\star - p}{L_s} v$ `// Eq. 4, |δw| ≤ Δ/2`
end
$\theta' \leftarrow \{\theta_\ell\}$ `// |s(θ')| = |s(θ)|: zero added memory`

`// Phase 2: restore, read time`

$\tilde{\theta} \leftarrow$ read from faulty memory;
$(v, d) \leftarrow \mathrm{PRNG}(\kappa)$
foreach *carrier group* $w' \subset \mathcal{C}$ **do**
  $\hat{\beta} \leftarrow \lfloor (w' \cdot v + d)/(\Delta/2) \rfloor \bmod 2$
end
unpack $\{\hat{\pi}_c\}$ from $\hat{\Pi}$
foreach *codeword* c *of* $\pi(\mathcal{P})$ **do**
  $\hat{M}_c \leftarrow \mathrm{BCH_Decode}(\tilde{M}_c, \hat{\pi}_c)$;
  $\hat{\theta}_c \leftarrow \hat{M}_c \cdot 2^{b-K} + (\tilde{\theta}_c \bmod 2^{b-K})$;
  flag if syndrome nonzero
end
return $\hat{\theta} \leftarrow \pi^{-1}(\{\hat{\theta}_c\}) \cup \{\tilde{\theta}_\ell\}_{\ell\notin\mathcal{P}}$

codeword is about $M^{-5} = 10^{-15}$. Master Table 6, block B, is the measured counterpart: 500 real PBS flips, 22 uncorrectable codewords under adjacent grouping and none under interleaving.

Sign-bit bound. A sign-bit attack alters one bit plane per weight; each weight contributes its sign bit to exactly one codeword, so the attack injects at most G errors per codeword. If $G \leq t$ every sign-bit configuration lies within the correction radius and is repaired exactly. Defeating a codeword therefore requires a second error on some weight, that is a second flipped plane. □

Proposition 3. At $L_s = 1$ the direction $v \in \{-1, +1\}$ is a scalar and $p = vw + d$ is an affine bijection of the integer weight. With Δ equal to one quantization step the two interleaved lattices are the even and odd integer levels, so snapping p to the lattice selected by β sets the parity of the integer level, that is the LSB of the stored weight. The dither $d = \frac{\Delta}{2} z$ shifts the lattice by half a step, which toggles which lattice is labelled β and complements the written bit when $z = 1$; the composition writes $\beta \oplus z$. Reading inverts the same affine map and takes the result modulo 2, recovering $\beta \oplus z$, and XOR with z returns β. Rounding to the nearest level tolerates any displacement below $\Delta/2$. □

D QUALITATIVE COMPARISON OF WEIGHT-PROTECTION SCHEMES

Table 10 places SANAI beside the schemes discussed in Section 2 along the axes that decide whether a protection is deployable on an edge accelerator: added memory, where the parity lives, the granularity

at which protection is applied, the fault tolerance actually demonstrated, the accuracy cost of the protection itself, whether retraining is needed, whether the parity is bound to a key, whether it survives fine-tuning, and which architectures it was shown on. Three patterns stand out. First, every scheme that adds no memory stores its parity next to the data it protects, so a burst that hits a block takes weight and check bits together; SANAI is the only one whose parity is moved cross-layer into carriers chosen for their tolerance. Second, no prior scheme binds parity to a key, so a copied model can always be repaired by whoever holds it; the key-seeded STDM embedding is what turns fault tolerance into an access-control primitive. Third, the demonstrated fault tolerance of the embedded schemes stops at a six percent BER, and none report behaviour under fine-tuning or on transformers or language models. The last row summarizes what the rest of the paper measures.

Table 10: Qualitative comparison of weight-protection schemes. * marks the strongest prior embedded scheme. SANAI is the only scheme that keeps parity in-weight and moves it cross-layer, decoupling the parity from what it protects.

Scheme	Mem.	Parity	Granul.	Fault tol.	Acc. drop	Retrain	Key	Fine-tune	Arch.
Conventional ECC (Reagen et al., 2018)	$\sim$12.5%	External	Word	High, 1 to 2 bit	Negligible	No	None	Unaffected	Agnostic
Weight scaling (Hong et al., 2019)	Low	None	Tensor	Low at high BER	$> 3\%$	No	None	Weakens	CNN
NN-ECC (Koppula et al., 2019)	Low	In-weight	Layer	High, learned	Low	Full	None	Re-embed	CNN
Weight throttling (He et al., 2020)	Low	None	Clipped	Saturates	Moderate	Yes	None	Drift-sensitive	CNN
Selective prot. (Rakin et al., 2019)	Low	External	Weight	Uneven	Mod. high BER	No	None	Not rep.	CNN
Weight nulling (Qin et al., 2017)	Low	None	Zeroing	Fails at 6%	High	No	None	Not rep.	CNN
Opportunistic (Burel et al., 2021)	Low	In-weight	Block	Gaps	High at 6%	No	None	Not rep.	CNN
Value-aware (Lee & Yang, 2022)	Low	In-weight	Magnitude	Medium	Moderate	No	None	Not rep.	CNN
E3-CODE* (Kamineni et al., 2026)	Low	In-weight	Layer-uniform	High to 6%	Low to 6%	No	None	Not rep.	CNN
SANAI (ours)	**Zero**	**Cross-layer**	**Adaptive BP**	**High, 10 to 12%**	**0.97 to 0.99**	**No**	**Key-gated**	**Parity BER 0**	**CNN, ViT, LLM**

E EXPERIMENTAL DETAILS

Table 11: Evaluated vision architectures. Params is the ImageNet reference count; the INT8 footprint in MB equals the parameter count in millions. CIFAR-adapted variants of the first three models (3×3 stride-1 stem, no early max-pooling) hold 11.16M, 23.52M, and 14.73M weights.

Model	Family	Params (M)	Datasets
ResNet-18 / ResNet-50 (He et al., 2016)	Residual CNN	11.7 / 25.6	C10, C100, IN
VGG-16 (Simonyan & Zisserman, 2015)	Plain CNN	138.4	C10, C100
MobileNet-V2 (Sandler et al., 2018)	Depthwise CNN	3.5	C10, C100, IN
EfficientNet-B0 (Tan & Le, 2019)	Depthwise CNN	5.3	C10, C100, IN
Wide ResNet-50-2 (Zagoruyko & Komodakis, 2016)	Residual CNN	68.8	IN
RegNet-Y-16GF (Radosavovic et al., 2020)	Design-space CNN	83.4	IN
Swin-T (Liu et al., 2021)	Hierarchical ViT	28.2	IN
MaxViT-T (Tu et al., 2022)	Hybrid ViT	30.4	IN
ViT-B/16 (Dosovitskiy et al., 2021)	Plain ViT	85.8	IN

Table 12: Operating points of the one mechanism. Both codes are shortened from parents over $\mathrm{GF}(2^7)$ ($x^7 + x + 1$, $n = 127$); "48+88" means 48 message plus 88 stored parity bits per codeword. G is weights per codeword, $f = t/n$ the correctable fraction of the parent, ρ the coverage. The sign-bit guarantee $G \leq t$ holds only for the LLM point. All points add zero memory.

Regime	Code	K	G	f	ρ	Interleave	Used in
Random BER, vision	strong: parent BCH$(127, 50, 12)$, 48+88 bits	3 (4 depthwise, ImageNet)	16 ($> t$)	9.4%	20% (40% ImageNet)	optional	Sec. 5.1, 5.2, 5.2
Attacks, vision	light: parent BCH$(127, 99, 4)$, 96+32 bits	4	24 ($> t$)	3.1%	10%	keyed	Sec. 5.3
LLMs	strong: parent BCH$(127, 50, 12)$, 48+88 bits	4	12 ($\leq t$)	9.4%	10 to 60%	keyed (attacks)	Sec. 5.4

Quantization and injection. PTQ uses per-output-channel scales for convolutions and per-tensor scales for linear layers. QAT checkpoints (Jacob, LSQ) store their exact training-time scales, so integer tensors are recovered losslessly before protection. Faults are injected into the two's-complement byte representation of the integer weights. Each BER point uses a fixed random seed for vision; LLM random-BER results average three seeds (0, 1, 2). CIFAR models are trained to convergence; ImageNet models use torchvision pretrained weights, with clean INT8 top-1 of 76.1 to 83.4%.

Sensitivity proxies. For vision the probe of Eq. 2 at $p_0 = 2\%$ ranks layers. For LLMs three proxies were compared: weight-ℓ_1 magnitude, gradient-weighted magnitude $\sum |g \odot w|$ from one backward pass over four 5-shot MMLU prompts, and the attack's own per-layer loss probe. The two gradient-informed proxies choose 118 small layers (39.1% of weights) on Qwen2-0.5B where

magnitude chooses 32 large ones (39.0%); Appendix J shows only the gradient-informed proxies protect the attacked layers. Token-embedding and output-projection matrices are excluded as carriers and from injection.

Attack re-implementations. PBS and T-BFA follow the public implementations (Rakin et al., 2019; 2021) on the INT8 tensors; T-BFA uses 90 source-target pairs on CIFAR-10 and 132 on ImageNet. AttentionBreaker follows the public code: the attack loss is the causal-LM cross-entropy over a batch of four 5-shot MMLU prompts; the per-weight score is $\alpha\widehat{|g|} + (1-\alpha)\widehat{|w|}$ with $\alpha = 0.5$ min-max normalized per layer; every attention and MLP projection is ranked by the loss reached after flipping the sign bit of its top $p\%$ weights, $p \in \{10^{-3}, 10^{-2}, 0.1, 1, 5\}$; the population space is the top 2% of the most fragile layer; GenBFA minimizes the flip set subject to $\mathcal{L} \geq 7$ with tournament selection, uniform crossover, 1 to 9% drop-mutation, and 100 generations of 40 candidates. All flips are on bit 7. We re-implement on SANAI's own INT8 tensors rather than on `bitsandbytes` LLM.int8() (Dettmers et al., 2022) so that every flip lands on exactly the integer SANAI encodes. MMLU for the attack experiments uses 500 questions from 20 subjects and WikiText-2 perplexity 20k test tokens.

F PORTABILITY TO A DEPLOYMENT FORMAT

To confirm that SANAI does not depend on the training framework, we apply it directly to ONNX graphs (Bai et al., 2019): the INT8 weight initializers of the exported model are the only inputs to the scheme, and the protected file runs unchanged in ONNX Runtime (ONNX Runtime developers, 2018) with no custom operator or runtime hook. On the CIFAR-10 SimpleCNN (static QDQ INT8, 88.15% clean), a BCH$(63, 39, t{=}4)$ code over GF(2^6), shorter than the paper's family to match the small classifier, protects the top four planes of the classifier, and its parity is hidden in the convolution layers by integer-lattice STDM ($\Delta = 8$, one bit per 16-weight block) at zero file growth (154,589 bytes before and after) and zero accuracy cost; protecting all eight planes costs 0.18 points. Twenty sign flips and a 1% random BER are fully detected (18 of 18 and 40 of 40 affected codewords flagged, none falsely) and corrected bit-exactly. On DistilGPT-2 (weight-only per-channel INT8, 76.4 WikiText-2 perplexity under this quantizer and token set, hence not directly comparable with Table 2), we protect the sign and top magnitude planes of the most gradient-sensitive projection (`h.0.attn.c_proj`) and, relaxing the carrier exclusion of Section 4, place the 884,736 parity bits in the token-embedding table, the largest and least sensitive tensor in the graph, again at zero file growth and a 0.04 perplexity cost. Fifty gradient-targeted sign flips that raise perplexity to 3,080 are flagged (38 of 38 affected codewords) and reversed exactly, and a 0.1% BER over all planes is brought from 258 to 77.7 perplexity, near clean, by restoring the two protected planes alone. In both cases every corrupted codeword was flagged and every clean codeword passed, so the sensitivity-aware, zero-overhead protection and the detect-and-flag behaviour carry over unchanged to a standard deployment format.

G STORAGE ACCOUNTING AND THE PRECISION SQUEEZE

Table 13: Left: storage cost at the efficient operating point (INT8). Middle: absolute counts in millions; [a]CIFAR-100. Right: fit for the ImageNet backbones, protected fraction, hidden parity as a fraction of model bits, parity bits per model weight, and carrier layers (insensitive / protected-layer overflow); the Swin-T, RegNet, and Wide ResNet rows are from the earlier logs pending re-extraction. Added memory is zero in every row.

Arch	Data	Cov.	b/wt
ResNet-18	C10	19.3%	1.06
ResNet-18	C100	14.0%	0.77
ResNet-50	C10	19.2%	1.05
VGG-16	C10	20.0%	1.10

Arch	Lay.	Wts	Bits	Prot. wts	Parity bits
ResNet-18	13/20	11.16	89.3	2.12 (19.0%)	11.64 (13.0%)
ResNet-18[a]	11/20	11.17	89.4	1.56 (14.0%)	8.62 (9.6%)
ResNet-50	28/53	23.52	188.2	4.52 (19.2%)	24.78 (13.2%)
VGG-16	10/16	14.73	117.8	2.95 (20.0%)	16.18 (13.7%)

Model	Prot.	Parity	b/wt	Carr.
MaxViT-T	40.9%	37.5%	3.00	132
Swin-T	39.5%	36.2%	2.90	35 / 0
RegNet-Y-16GF	40.0%	36.6%	2.93	70 / 1
Wide RN-50-2	37.3%	34.2%	2.73	21 / 0
ViT-B/16 ($K{=}5$)	43.5%	53.2%	4.26	26 / 8

Counting the full code expansion $1 + p/k_s = 1 + 88/48$ instead of the systematic parity fraction p/k_s would predict about 1.7 bits per weight at the default operating point and contradict the measured 1.05 to 1.10; the systematic property is what keeps the cost low. For the LLM random-BER experiments, all three small models hide the full payload at 3.5 to 4.0 bits per weight from 17 to 32 insensitive carrier layers (Qwen2-0.5B: 24 insensitive plus 8 overflow; GPT-2: 32 plus 8; DistilGPT-2: 17 plus 1), and the attack-regime configuration at 10% coverage uses about 9.2% of model bits with no overflow.

Precision squeeze. Two effects act together as precision falls. Lower precision makes more layers fragile, so the sensitive set grows and demand rises; the same shrinking bit budget leaves fewer robust carriers with spare low-order bits, so the capacity $W_c h$ falls. Figure 3 (right) plots both: demand rises from about 13% of model bits at INT8 to about 70% at INT4 while capacity falls from about 62% to 18%, and past the crossing no assignment of parity to carriers satisfies Eq. 6. INT6 on CIFAR-10 remains feasible (a small budget degrades to about 15% at 12% BER, 35% coverage with $K = 4$ recovers to about 86%); INT6 on ImageNet and INT4 sit past the crossing.

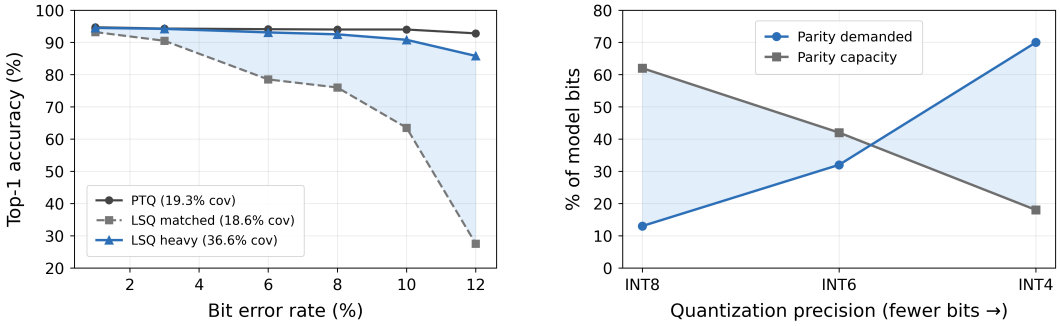


Figure 3: Left: quantizer ablation; LSQ needs roughly 2.5× the protection budget of PTQ. Right: the precision squeeze; parity demand rises while carrier capacity falls, and configurations past the crossing cannot be protected at zero added memory.

H Extended Vision Results

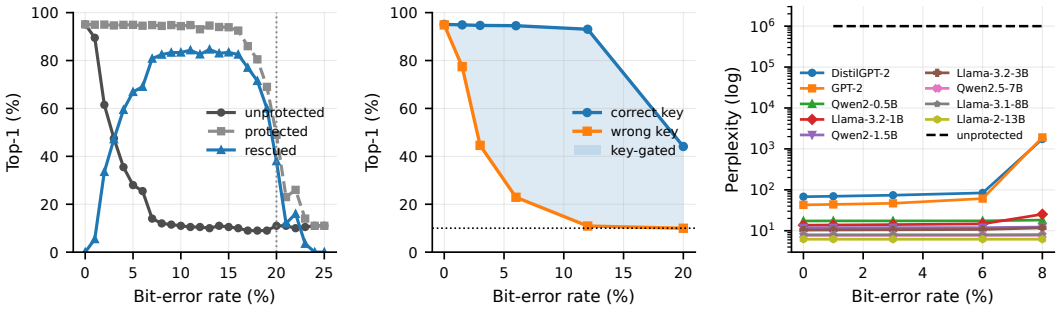


Figure 4: Left: unprotected, protected, and rescued accuracy over a 0 to 25% BER sweep (ResNet-18, CIFAR-10); the rescue plateaus and falls only near the code limit at 20%. Middle: recovery with the correct versus a wrong key; the shaded gap is recoverable only with the key. Right: protected LLM perplexity versus BER on WikiText-2 (log scale); the unprotected model sit at the 10^6 ceiling.

Table 14: Full comparison with embedded ECC under burst errors: top-1 accuracy at 1.5, 3, and 6% BER for all six methods and eight settings. Norm. is accuracy normalized to each method's own software baseline; baseline values as reported by Kamineni et al. (2026).

		SW	W. Nulling			Opp. Parity			Value-Aware			E3-CODE			E3-CODE-H			SANAI (ours)		
Data	**Model**	**Acc.**	1.5	3	6	1.5	3	6	1.5	3	6	1.5	3	6	1.5	3	6	1.5	3	6
C10	ResNet-18	82.0	82.0	19.0	0.05	81.7	81.4	10.9	82.0	81.2	77.5	81.8	81.8	78.8	82.0	81.9	81.3	94.9	94.7	**94.5**
	ResNet-50	85.2	85.0	54.1	5.5	84.8	84.0	9.1	85.0	84.7	79.5	85.1	84.6	83.4	85.1	84.8	84.0	94.9	94.8	**94.5**
	MobileNet-V2	83.6	83.0	8.8	0.0	82.9	79.5	9.1	83.0	81.3	41.2	83.0	82.4	59.4	83.6	83.0	76.8	78.2	76.6	75.3
	EfficientNet-B0	82.8	82.0	2.9	0.0	80.3	64.8	9.8	82.0	77.9	22.6	82.3	78.6	55.6	82.7	82.3	80.1	83.8	83.0	**81.4**
IN	ResNet-18	69.5	69.3	10.0	0.4	69.2	67.3	0.1	69.3	68.2	59.0	69.3	68.9	65.3	69.4	69.3	67.7	75.5	75.1	**73.3**
	ResNet-50	76.1	75.9	18.8	0.4	75.8	73.8	0.0	75.9	75.1	65.1	75.9	75.8	70.0	76.1	75.9	74.7	82.2	81.6	**80.5**
	MobileNet-V2	71.5	69.9	5.5	0.0	65.8	58.1	0.1	69.9	66.0	20.9	69.9	66.7	36.9	71.2	70.5	65.5	78.4	78.1	**77.8**
	EfficientNet-B0	77.0	73.7	1.1	0.0	74.1	48.8	0.1	73.7	56.8	0.5	75.7	70.9	1.8	76.6	75.5	64.0	82.5	80.6	**76.2**
Norm.		1.00	0.98	0.19	0.01	0.97	0.88	0.06	0.98	0.94	0.58	0.99	0.97	0.72	0.99	0.99	0.94	**0.99**	**0.99**	**0.97**

Critical BER. Figure 5 reports the improvement in critical BER relative to the unprotected model on ResNet-50/ImageNet. SANAI reaches 6×10^{-2}, 300,000× the unprotected 2×10^{-7} and 4× beyond E3-CODE-H at 1.5×10^{-2}. The same ordering holds on EfficientNet-B0, where our 3×10^{-2} exceeds E3-CODE-H at 7×10^{-3} and value-aware parity at 4×10^{-4}. Under random BER at the attack operating point (rows V2.9 to V2.13 of Master Table 5) the compact architectures hold through 9%

but drop at 12%, consistent with the lower $f = 3.1\%$ of the short $t = 4$ code; the long $t = 12$ code of the random-BER operating point is the right choice for that regime, which is why Table 12 keeps both.

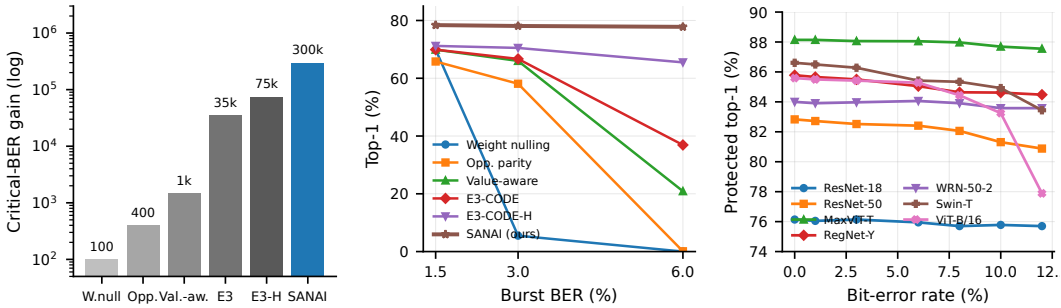


Figure 5: Left: critical-BER improvement ratio over the unprotected model, ResNet-50 on ImageNet (log scale); baseline values as reported by Kamineni et al. (2026). Middle: MobileNet-V2 on ImageNet under the burst protocol against prior embedded ECC; the strongest prior method bends downward while ours stays flat. Right: modern architectures over a 0 to 10% sweep (ImageNet, INT8); ViT-B/16 omitted for scale.

I Extended Attack Results on Vision Models

Figure 6 illustrates the mechanism; Figure 7 shows the measured per-codeword fault map behind Table 1 and Figure 8 the accuracy across all six settings. An independent fault-map pass over the same ResNet-18/CIFAR-10 attack, counting only faults in the four protected planes with its own weight selection, gives the same picture with larger counts (29 uncorrectable codewords, the worst with 27 faults; 0 and a maximum of 3 after interleaving) and adds three observations: the flips concentrate on bits 7 and 6 and almost never touch bit 4, so the attack targets exactly the planes the code protects; the failing codewords sit in `conv1`, `layer1`, and `layer2`, where PBS focuses; and the worst codeword fails because nearly every one of its 24 weights is hit on bit 6 or 7, the concentration that the keyed permutation breaks (Figure 9). Under adjacent grouping at 500 PBS flips on ResNet-18/CIFAR-10, 22 codewords exceed $t = 4$ with a peak of 14; under interleaving the identical faults spread across many codewords, none exceeding 3. Across the six settings (Master Table 6, block B) adjacent grouping leaves 9 to 49 codewords uncorrectable and accuracy at 0 to 35%, while keyed interleaving leaves none (five on ResNet-50 ImageNet) and restores 94.9, 94.8, 75.5, 75.8, and 81.9%. ResNet-18 on ImageNet is the one setting where interleaving corrects every codeword yet accuracy does not return (1.5%): at 10% coverage and 500 flips the unprotected 90% of the model is corrupted, so correcting the protected codewords is insufficient; higher coverage or a lower flip budget resolves it. On the compact networks (Master Table 6, block D) interleaving removes 90 to 100% of the uncorrectable codewords (42 to 1, 35 to 0, 17 to 0, 23 to 3) and cuts the peak fault count from 10 to 24 down to 2 to 8; the residual one to three codewords on MobileNet-V2 and EfficientNet-B0 are all in the classifier layer, on which the attack concentrates entirely. The bit-level map of the most heavily hit interleaved codewords (Figure 9) shows no weight carrying more than one flip. T-BFA succeeds almost always unprotected (90 to 100% ASR with 10 to 20 flips), 97% of which land on `fc.weight`; under SANAI the ASR falls to 0% on most of the 132 ImageNet pairs, with a few residual leaks (Figure 11).

Attack-strength sweep on ImageNet: interleaving versus coverage. To separate the effect of keyed interleaving from the effect of coverage, we sweep the PBS attack on ResNet-18/ImageNet from 10 to 500 flips (light code, $K = 4$, 10% coverage) and record, for adjacent and keyed grouping, the protected accuracy and the largest fault count in any codeword (Table 15, Figure 10). Two things follow. First, interleaving never fails at the codeword level: the interleaved maximum stays at 1 or 2 faults, below $t = 4$, and no codeword becomes uncorrectable at any strength, whereas under adjacent grouping the maximum passes t by 40 flips, reaches 12 by 250, and up to 14 codewords are uncorrectable. Second, interleaving extends the usable range substantially: at 100 flips adjacent grouping has fallen to 46.0% while interleaved is at 75.4%, and at 250 flips adjacent is at 0% while interleaved retains 50.6%. The interleaved accuracy does decline at very high budgets (1.5% at 500 flips), but at those points every protected codeword is still corrected; the loss comes from the 90% of the model left unprotected at 10% coverage, which the attack corrupts directly once the budget is

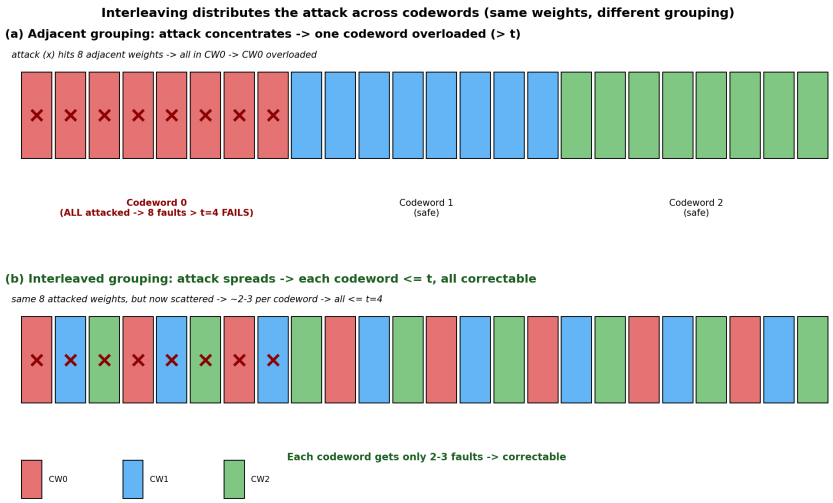


Figure 6: Interleaving mechanism. Top, adjacent grouping: the attacked weights fall into one codeword and overload it ($f_c > t$). Bottom, keyed interleaving: the same weights are scattered by the keyed permutation so that each codeword holds at most a few faults, all within t. The measured fault maps for a real 500-flip attack are in Figures 7 and 9.

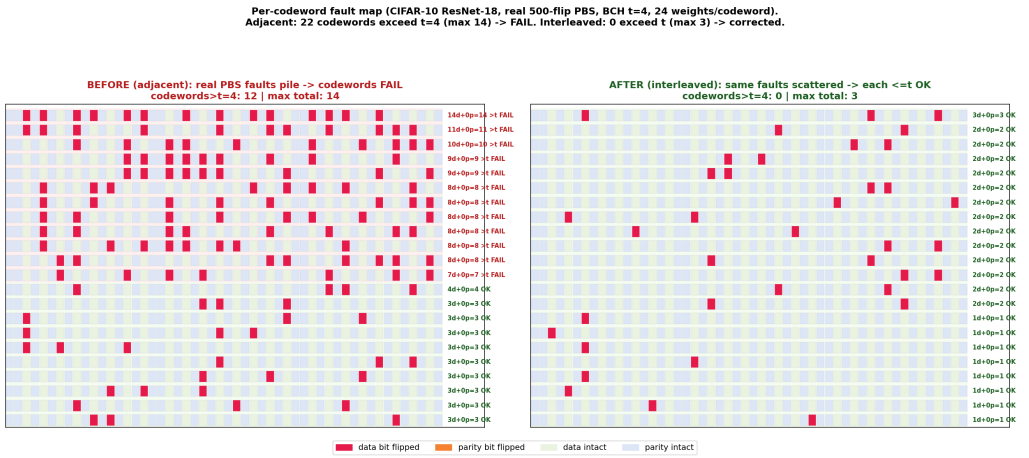


Figure 7: Per-codeword fault map under a real 500-flip PBS attack (ResNet-18, CIFAR-10, light code $t = 4$, 24 weights per codeword). Each row is one codeword and each red cell a flipped bit; the panels show the most heavily hit codewords. Left, adjacent grouping: the attack piles faults into a few codewords, 22 of which exceed $t = 4$ (up to 14 faults) and fail. Right, keyed interleaving: the identical faults are scattered by the keyed permutation so that no codeword exceeds 3 faults and every codeword is corrected. The attacker flipped the same bits in both panels.

large. On this model the bottleneck at high attack strength is therefore coverage, not interleaving, which is why the ImageNet operating point of Section 5.1 uses 40%.

J Extended LLM Results

Figure 12 plots the protected MMLU and ARC-Easy accuracy of Table 7 against BER: every protected model stays well above the 25% chance line through 6%, while the unprotected models fall onto it at the first nonzero BER; the 6% points of the 7B and 8B are conservative for the overflow reason given in the table caption.

Layer ranking and coverage. On Qwen2-0.5B under 10 sign flips (undefended 25.8% / 1,294), a weight-ℓ_1 ranking leaves the attacked `layers.1.self_attn.o_proj` unprotected and the model collapses as if undefended (row L2.8 of Master Table 8), whereas gradient-weighted magnitude and the attack-probe loss both select it and recover 45.6% / 17.65 at every coverage from 10% to

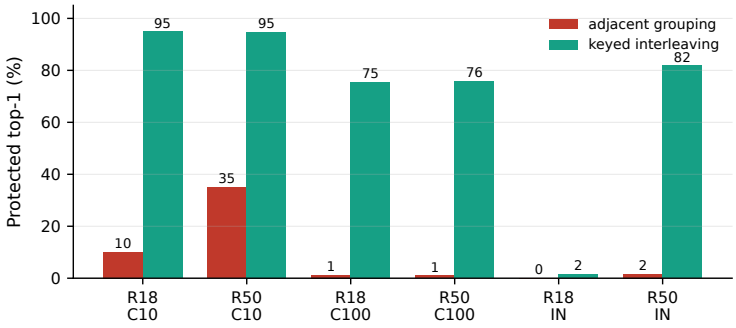


Figure 8: Protected top-1 accuracy under the 500-flip PBS attack, adjacent grouping versus keyed interleaving, across six architecture and dataset settings (light code, $K = 4$, 10% coverage). Interleaving restores near-clean accuracy in every case except the coverage-limited ResNet-18 on ImageNet.

Table 15: ResNet-18/ImageNet PBS flip sweep (INT8, light code $t = 4$, $K = 4$, 10% coverage): protected top-1 (%), largest fault count in one codeword, and number of uncorrectable codewords, adjacent versus keyed grouping, same flipped bits.

	Top-1 (%)		**Max faults / cw**		**Codewords** $> t$	
Flips	Adj.	Keyed	Adj.	Keyed	Adj.	Keyed
10	76.14	76.14	1	1	0	0
20	76.14	76.14	3	1	0	0
40	74.19	76.08	5	1	2	0
50	73.81	76.06	5	1	2	0
100	45.98	**75.41**	8	1	3	0
150	11.88	**70.55**	9	2	3	0
250	0.00	**50.55**	12	2	10	0
350	0.00	22.34	12	2	11	0
450	0.00	7.12	12	2	14	0
500	0.00	1.53	12	2	14	0

40% and under both layouts (rows L2.2, L2.9). On Llama-3-8B-Instruct the attack concentrates on `layers.1.mlp.down_proj`; at 10% and 20% coverage that layer sits just outside the selected set and the recovered MMLU stays at the attacked 27.0, at 30% it enters and recovery is exact (71.0; rows L2.23 to L2.25). This is a coverage property, not a code failure: whenever the attacked layer is protected, recovery is bit-exact, and the minimum coverage that guarantees this grows with model size.

Code-aware attacker on Llama-3.2-1B. The 1B model yields little to the code-aware attacker (rows L3.4 and L3.5 of Master Table 9): 1,300 two-plane errors on the adjacent layout raise the corrected perplexity only from 14.20 to 23.0, the blind attacker on the interleaved layout achieves nothing, and the keyed attacker needs 2,600 errors to reach 129.5.

Qualitative outputs. Table 16 shows generated answers from Llama-3-8B-Instruct in four states. AttentionBreaker collapses generation into repeated-token garbage; SilentStriker keeps the answer fluent but changes it, dropping the correct option or drifting off the question; after SANAI the output is bit-identical to the clean model in both cases, because the restored weights are bit-identical to clean. Qwen2-0.5B behaves the same way under the 100-flip attack: the attacked model emits code fragments, digit runs, or characters from another script, and after SANAI the generation is the clean model's generation again.

K FINE-TUNING DETAILS

Protocol. Because the parity lives in the low-order bits of carrier weights, any later update of the model could wash it out. We protect ResNet-18 and ResNet-50 on CIFAR-10 at the random-BER operating point (strong code, $K = 3$, 19.3% and 19.2% coverage, $L_s = 1$), then fine-tune the protected INT8 model for ten epochs with SGD at three learning rates, 10^{-4}, 10^{-3}, and 10^{-2}, re-quantizing after every epoch. After each epoch we regenerate the key-seeded projection, de-project the parity from the drifted carriers, and compare it bit by bit with the parity written at protection time;

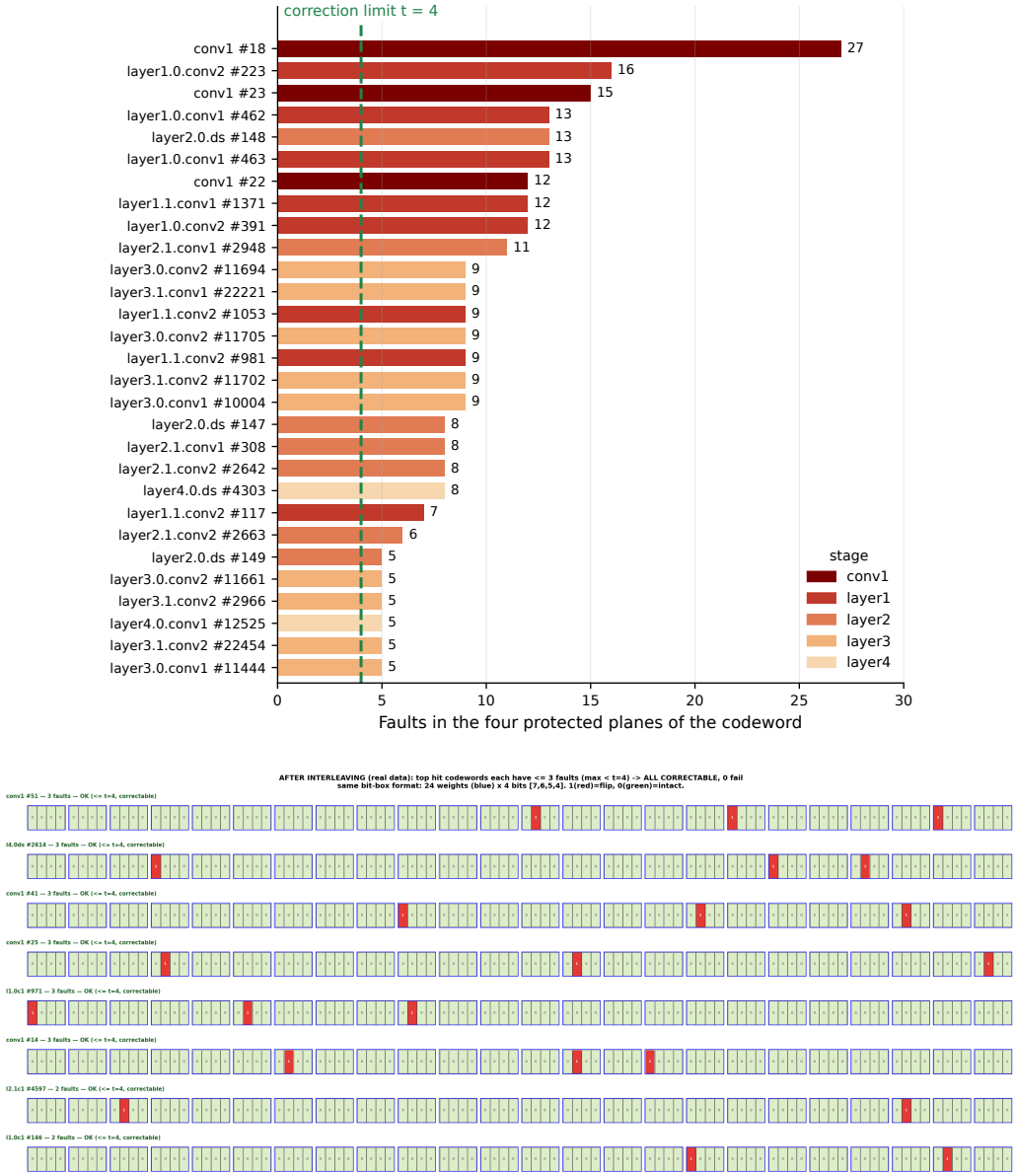


Figure 9: Fault-map pass over the 500-flip PBS attack on ResNet-18/CIFAR-10 (light code, $t = 4$, 24 weights per codeword, faults counted in the four protected planes). Top: all 29 codewords that exceed t under adjacent grouping, ranked by fault count and labelled by layer and codeword index; the worst, `conv1` #18, takes 27 faults, and the failures sit in the early convolutions where PBS concentrates. Bottom: bit-level map of the most heavily hit codewords after keyed interleaving; each row is a codeword of 24 weights with its four protected bits (7, 6, 5, 4), a highlighted cell is a flipped bit. Faults are scattered so that no weight carries more than one flip and no codeword holds more than 3 faults, all correctable.

the parity BER is the fraction of parity bits that no longer decode. We also record the fine-tuned model's clean accuracy and, at the end of training, inject a 12% BER into the protected layers and decode with the surviving parity.

Findings. At the two learning rates used for realistic adaptation, 10^{-4} and 10^{-3}, the parity is bit-exact after all ten epochs on both architectures (parity BER 0.000%) and accuracy stays near 95% (Table 17). Only at the aggressive 10^{-2} rate does any parity move, and then late and slightly: 0.003% of parity bits on ResNet-18 and 0.004% on ResNet-50 by epoch ten, at which point the model itself has lost six to seven points of accuracy to the large steps. The drift is confined to the earliest convolutional layer, whose weights move past the STDM tolerance $\Delta/2$ under large updates

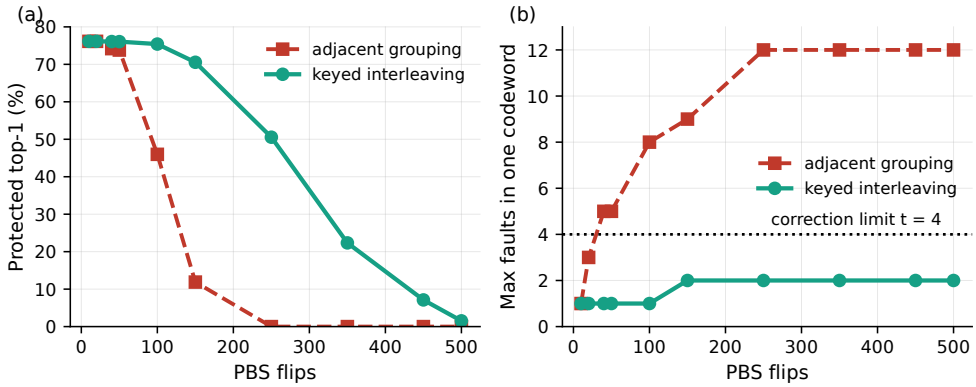


Figure 10: ResNet-18/ImageNet flip sweep. Left: protected accuracy versus PBS flips; keyed interleaving holds far longer than adjacent grouping. Right: largest fault count in one codeword; interleaving stays at 1 to 2, below $t = 4$, at every attack strength while adjacent grouping rises to 12. Where the interleaved accuracy still drops at high flip counts the per-codeword count is still below t, so the loss comes from the unprotected fraction (a coverage limit), not from interleaving.

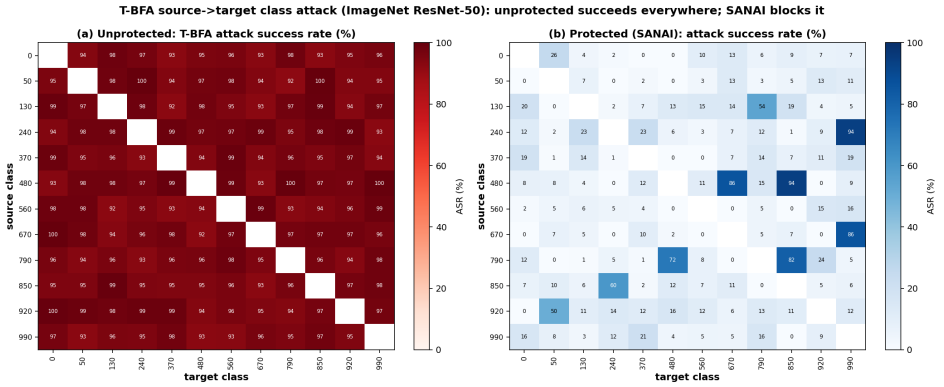


Figure 11: T-BFA source-to-target attack matrix (ImageNet ResNet-50, 132 source-target pairs). Each cell is the attack success rate for forcing the row (source) class to the column (target) class. (a) Unprotected: the attack succeeds on essentially every pair (90 to 100%). (b) Protected (SANAI): the success rate collapses to near zero on almost all pairs, with a few residual leaks, because the flips land in the protected classifier weights and are corrected before inference.

(Figure 13, right); every other carrier stays within tolerance. The surviving parity remains usable: correcting a 12% BER with it holds 93.8% (ResNet-18) and 94.7% (ResNet-50), so a protected model can be adapted in the field without re-embedding. Because the drift is localized to one early layer, the remedy is simple, either exclude the earliest layers from the carrier set or give them a wider lattice step, at a small cost in capacity.

L Ablation Tables

Uniform fault model: interleaving protects the parity under carrier corruption. The parity is interleaved along with the data: the keyed permutation scatters each codeword's parity bits across the carrier layers. When the carrier layers themselves are corrupted at the full fault rate, that corruption is therefore spread over many codewords, and each codeword's parity stays within the correction budget t. Table 18 shows this on ResNet-18/CIFAR-10 (strong code, $t = 12$, $K = 3$, 19% coverage) with the carrier layers corrupted at the same BER as the protected weights, which is the uniform model of Section 7. At 12% BER, interleaved parity keeps protected accuracy at 89.3% against 78.8% with sequential parity placement, while the unprotected model is at 10.8%; at 6% both placements hold near 94%. Keyed interleaving thus protects both the data and the parity, and the flat curves of Section 5.1 do not depend on clean carriers, only on the parity being scattered (rows V1.36 and V1.37 of Master Table 4).

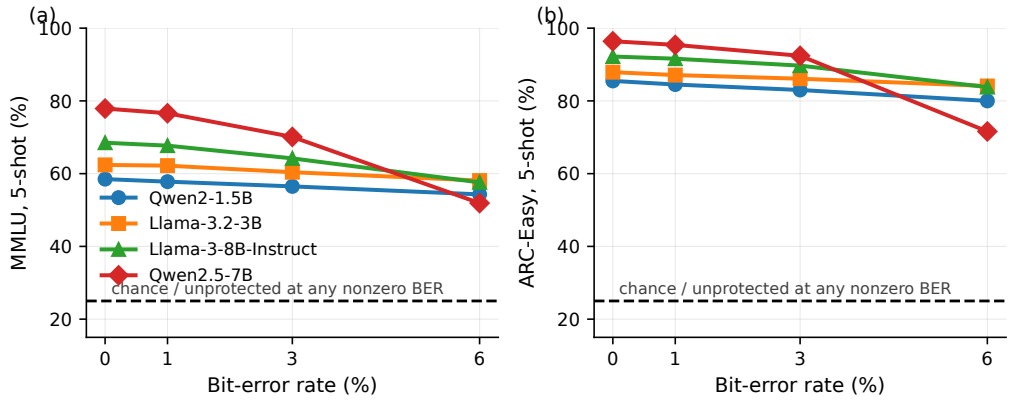


Figure 12: Protected accuracy versus BER on (a) MMLU and (b) ARC-Easy, 5-shot, 2,000 questions, three seeds; every protected model stays well above chance ($\approx 25\%$), whereas unprotected models fall onto the chance line at any nonzero BER.

Table 16: Representative Llama-3-8B-Instruct INT8 outputs before attack, under AttentionBreaker and SilentStriker, and after SANAI recovery. AttentionBreaker causes repeated-token corruption, while SilentStriker preserves fluency but alters the response. SANAI restores the clean output exactly.

Prompt	Before attack	AttentionBreaker	SilentStriker	After SANAI
What is true for a type-Ia supernova?	A type-Ia supernova is a specific astronomical event. Here are the options: A) explosion of a massive star; B) merger of two white dwarfs; C) collapse of a neutron star; …	`<repeated-token garbage; paste literal excerpt>`	A supernova is the explosive explosion of a star that has reached the end of its life, releasing a massive amount of energy …	A type-Ia supernova is a specific astronomical event. Here are the options: A) explosion of a massive star; B) merger of two white dwarfs; C) collapse of a neutron star; …
If you know both the actual and apparent brightness of an object, what can you estimate?	The correct option is: **Distance**. Apparent brightness is inversely proportional to the square of the distance …	`<repeated-token garbage; paste literal excerpt>`	I'm happy to help with any question you have. Perhaps you could use the above response as guidance or additional information …	The correct option is: **Distance**. Apparent brightness is inversely proportional to the square of the distance …
Why is the sky blue?	The correct answer is: **Scattering of light**. The sky appears blue because sunlight is scattered by molecules in Earth's atmosphere such as nitrogen and oxygen …	`<repeated-token garbage; paste literal excerpt>`	The sky is blue because of the scattering of sunlight. When sunlight enters Earth's atmosphere, it is deflected by tiny gas molecules …	The correct answer is: **Scattering of light**. The sky appears blue because sunlight is scattered by molecules in Earth's atmosphere such as nitrogen and oxygen …

Table 17: Parity survival under fine-tuning (CIFAR-10): parity BER after ten epochs and model accuracy, three learning rates. Bit-exact at realistic rates on both architectures.

Model	LR	Parity BER	Acc.	Model	LR	Parity BER	Acc.
ResNet-18	10^{-4}	0.000%	95.0%	ResNet-50	10^{-4}	0.000%	94.9%
	10^{-3}	0.000%	94.9%		10^{-3}	0.000%	94.8%
	10^{-2}	0.003%	88.5%		10^{-2}	0.004%	90.0%

Host capacity and embedding. At $L_s = 1$ STDM matches QIM capacity while adding key security; raising L_s spreads each bit over more carriers, improving robustness of the hidden parity at reduced capacity subject to Eq. 6. A wrong key yields approximately 50% parity recovery, confirming Proposition 2(b). Replication and coverage draw on the same carrier space, so higher R is paid for with lower coverage; the gain at the top of the range is modest and $R = 1$ retains the widest coverage, so it is the default and R is a knob for when extreme-BER robustness matters more than coverage.

M HARDWARE DETAILS

All numbers in this appendix are measured on the platform below unless marked modeled. Tool: Synopsys Design Compiler V-2023.12-SP5, gscl45nm standard cells, 100 MHz constraint; memories black-boxed as memory-compiler IP; power annotated from switching activity captured over a full-network RTL inference and back-annotated onto the compiled netlist. The library ships only

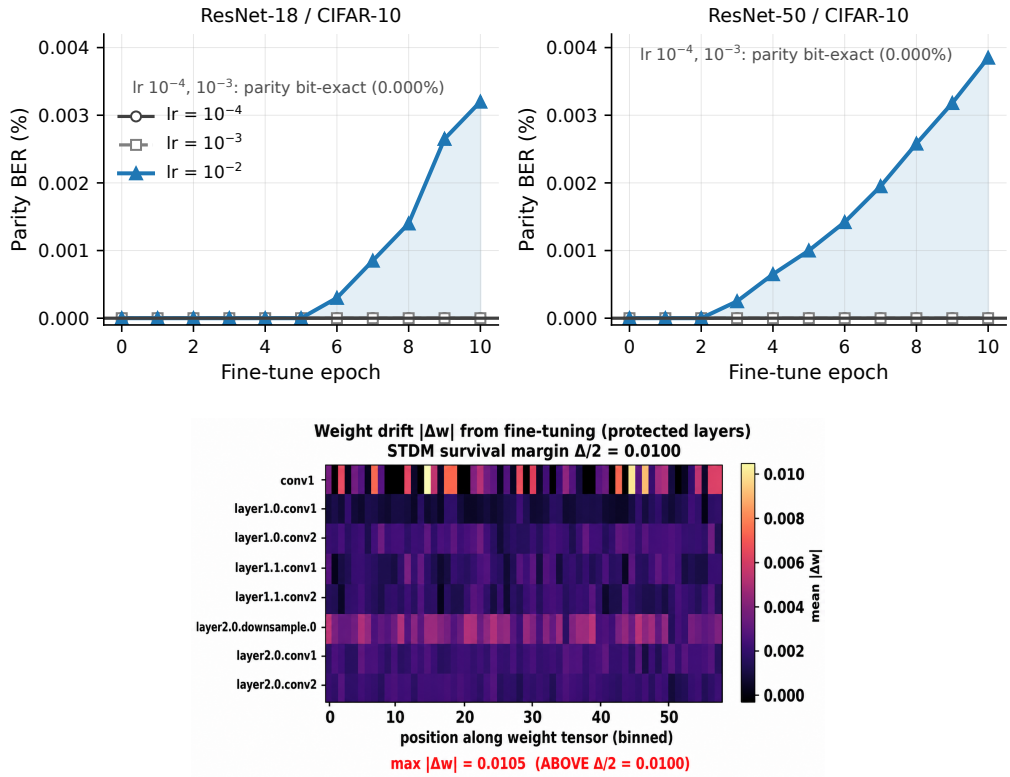


Figure 13: Fine-tuning. Top: fraction of parity bits that no longer decode after each epoch on ResNet-18 (left) and ResNet-50 (right); at learning rates 10^{-4} and 10^{-3} the parity stays bit-exact for all ten epochs, and only at 10^{-2} does it drift, late and slightly (0.003 to 0.004% at epoch ten). Bottom: per-layer weight drift after fine-tuning at 10^{-2} against the STDM tolerance $\Delta/2$; only the earliest convolution crosses it.

Table 18: Protected top-1 (%) under a uniform fault model that also corrupts the parity-carrying carrier layers (ResNet-18/CIFAR-10, strong code $t = 12$, $K = 3$, 19% coverage). Interleaving the parity keeps it correctable.

BER	Interleaved parity	Sequential parity	Unprotected
0%	95.06	95.06	95.06
6%	93.92	93.83	10.25
12%	**89.34**	78.83	10.78

Table 19: Ablations on ResNet-18 unless noted. Top left: coverage (CIFAR-10). Top right: code strength on ResNet-50/ImageNet at 23.4% coverage; the fraction $f = t/n$, not t, governs where a code fails. Bottom left: bit-plane depth (CIFAR-10, 40% coverage). Bottom right: parity replication on ResNet-50/CIFAR-10; each R uses the largest coverage that fits at zero added memory, and b/wt is parity per protected weight.

Coverage	Parity % bits	@ 6%	@ 12%
19.3%	13.26	94.54	93.06
52.8%	36.30	95.02	95.02

K (planes)	Knee BER	@ 16%	@ 20%
3	∼13%	84.1	24.5
4	∼16%	92.3	44.3

t	m	n (parent)	$f = t/n$	@ 6%	@ 10%	@ 12%
4	7	127	3.1%	2.6	0.0	0.1
8	7	127	6.3%	78.4	33.3	9.2
12	7	127	9.4%	80.7	78.9	77.3
20	8	255	7.8%	80.9	78.1	75.7

R	Cov.	b/wt	0%	6%	10%	12%
1	39.8%	5.50	94.86	94.97	94.75	94.84
2	20.4%	11.00	94.86	94.86	94.83	94.80
3	15.1%	16.50	94.86	94.95	94.97	94.78

a compiled `.db` without Verilog cell models, so gate-level SDF simulation was not possible and power figures are directional rather than SDF-accurate.

Platform. The accelerator is an 8×8 output-stationary systolic array in the style of the TPU (Jouppi et al., 2017), using the output-stationary dataflow of the Eyeriss taxonomy (Chen et al., 2016). Each

processing element holds a partial sum in place while INT8 weights are broadcast down columns and activations propagate east; convolutions are lowered to matrix multiplies via im2col. It runs a quantized CNN with five convolutional layers, two max-pool stages, global average pooling, and one fully-connected layer (140,384 INT8 weights) on CIFAR-10. Weights are held in on-chip ROM and streamed into the array under a per-layer finite state machine (Figure 14). The checker is a read-only tap on this path: it observes `w_rd`, `w_addr`, and `w_din` between the weight memory and the convolution engine, and neither the weight memory nor the PE array is modified.

Threat model. The adversary may write to weight storage but cannot read the on-chip seed. We target sparse bit flips of the kind produced by RowHammer, progressive bit search, and laser fault injection (Kim et al., 2014; Rakin et al., 2019; Skorobogatov & Anderson, 2003). Burst errors are outside the hardware scope; the software path of Section 5.1 covers that regime.

Two carriers. *Design A, the primary design,* is an LSB-parity gather. Each block of $L = 16$ consecutive weights forms a BCH$(126, 112, t{=}2)$ codeword over $\mathrm{GF}(2^7)$ with primitive polynomial $x^7 + x^3 + 1$ and generator $g(x) = \texttt{0x4377}$. The payload is the seven high-order bits of each weight (112 bits); the 14 parity bits occupy bit 0 of weights 0 through 13, leaving two spare LSBs untouched. The 140,384 weights form 8,774 blocks, one codeword each. *Design B* is the spread-transform variant. Each block is projected onto a pseudorandom ± 1 spreading sequence, $P = \sum_{i=0}^{L-1} w_i s_i$, and the projection is quantized on an integer lattice of step $\Delta = 8$, $q = \lfloor (P + D/2)/D \rfloor$ with $D = L\Delta/2 = 64$, carrying one bit per block as $q \bmod 2$. Because D is a power of two the recovery path reduces to an accumulator, an adder, and a shift, $q = (P + 32) \gg 6$; only the offline embedder performs the fractional nudge that moves P to the lattice point of the target bit. Blocks are grouped into BCH$(127, 113, t{=}2)$ codewords, 73 in total, split per layer with shortened tails so that each codeword lies within one keystream reseed interval. In both designs each parity bit is XORed with a keystream from a degree-20 LFSR ($x^{20} + x^{13} + x^9 + x^5 + 1$) seeded from a fuse-resident secret; without whitening an adversary could alter the payload, recompute the code, and rewrite the parity. The cost is one XOR gate. Design B uses a second, distinct seed for the spreading sequence. An LFSR keystream is linearly predictable from roughly twice its degree of observed output, unlike a cryptographic generator; an adversary would additionally require the projection values, which are not exposed, and a lightweight block cipher in counter mode (Borghoff et al., 2012) would close this gap at higher area cost. The hardware codes are independent of the software strong code of Section 3.3 (different field polynomial, different payload); the two embeddings coexist in the same weights and the hardware checker never reads the software parity.

Contiguous blocks and address keying. A streaming checker observes one weight per cycle and has no notion of global position, so a magnitude-selected scattered pool is unavailable to it; blocks are therefore $L = 16$ consecutive weights in fetch order. Stealth is not a requirement: an adversary who knows where the parity lives still cannot forge it without the seed. Block boundaries are keyed on `w_addr` (block $=$ addr $\gg 4$) rather than on counting fetch pulses. Address traces captured from the convolution engine show per-output-channel tile-boundary duplicate reads and channel-padding over-reads in the fully-connected layer, either of which would desynchronize a pulse counter; addresses beyond the layer weight count are ignored, and for the same reason the LFSRs advance on address increment rather than on every fetch pulse. The projection accumulator in Design B stores a per-position product rather than a running sum, making duplicate reads idempotent.

Detect-and-flag, not correct. Both deployed checkers detect and flag; recovery is performed in software from the parity embedded in the weights. We evaluated an in-hardware correcting variant and rejected it for two measured reasons. First, per-block correction cannot meet the one-cycle weight-delivery contract without a shadow weight memory of 140,384 bytes, equal to the entire weight file, which would eliminate the zero-storage property. Second, above the correction capacity the decoder miscorrects rather than flags: 46.99% of blocks at $t{+}1$ and 47.21% at $t{+}2$ for Design A, 43.83% and 46.10% for Design B, in each case silently writing incorrect weights. The checker is clock-gated on the weight-fetch condition; over a full inference of 4,444,928 cycles only 140,384 are weight-fetch cycles, a duty cycle of 3.16%.

Functional transparency. With no faults injected, both protected accelerators reproduce the reference integer datapath exactly: 100 of 100 images match the golden fully-connected logits

bit for bit, argmax agrees on every image, and no codeword raises a false flag. On the same full-data model and the same 100 images, top-1 is 86 unprotected, 88 under Design A, and 89 under Design B; the spread of three images is sampling noise at $n = 100$, consistent with a 1,000-image sweep that measures zero accuracy change from embedding. Running either checker against unembedded weights raises the flag on all 100 images, confirming that the zero-false-flag result is not a degenerate always-low output.

Detection. Table 20 reports measured per-bit detection for both designs; each entry is over 8,774 injected faults, one per block, for 70,192 faults per design. Design A was measured twice, by the software model and by RTL simulation, in agreement. Design A is magnitude independent: its gather reads the weight bits directly into the 126-bit codeword, so any single flip is a weight-one codeword error and $d_{\min} = 5$ detects it with certainty. Design B is magnitude dependent, a direct consequence of lattice quantization: a flip is detected only if it displaces the block projection across a lattice boundary, so high-order flips are detected with certainty while low-order flips that change a weight by a fraction of its scale are largely invisible and also harmless to accuracy. The sign bit is a structural blind spot for Design B: a sign flip displaces a weight by exactly $128 = 2D$, leaving the lattice parity unchanged, which follows from D being a power of two. We measured alternatives: $\Delta = 16$ closes the blind spot but halves detection at bit 5 (0.50 to 0.24) and bit 6 (1.00 to 0.52) while doubling embedding distortion; an odd Δ preserves the curve but lifts bit 7 only to about 21% and replaces the shift with a constant divider. We retain $\Delta = 8$ for Design B and take Design A as primary. At the codeword level both codes have $d_{\min} = 5$, so any pattern of one to four corrupted codeword bits produces a nonzero syndrome; an escape requires five or more corrupted bits forming a valid codeword.

Table 20: Measured block-level detection rate by flipped bit position (8,774 blocks per position, 70,192 injected faults per design).

Flipped bit	7 (sign)	6	5	4	3	2	1	0
Design A (%)	100	100	100	100	100	100	100	100
Design B (%)	0.0	100	50.4	23.4	11.3	6.1	2.9	1.4

Area, timing, and power. Table 21 gives the complete measurement. Design A adds 0.41% of logic area, 0.63% of dynamic power, and 0.07% of leakage, and no stall cycles: the flag is advisory and never gates the datapath. Clock gating costs 61.0 μm^2 for Design A and 89.2 μm^2 for Design B and does not regress function, verified by re-running the full regression set on the gated designs. The full-top critical path lengthens by 0.76 ns for Design A and 0.61 ns for Design B; neither is a checker path. In every configuration the critical path lies entirely inside the convolution engine (`cur_layer_reg` to `Wb_reg`), and the checkers close at 1.27 and 1.31 ns, more than $20\times$ faster. The increase is the synthesizer re-buffering the convolution-engine control chain under the tap's additional output load, a pre-layout perturbation recoverable at place-and-route or by registering the tap inputs. The unprotected accelerator itself does not close at 100 MHz; all configurations meet about 33 MHz, a constraint inherited from the baseline. Design A and Design B were synthesized in separate sessions with the same tool version, library, and constraint; the two clean baselines differ by 5.64 μm^2 (0.0004%). The full-top area delta for Design B is $+18{,}535$ μm^2, of which the checker instance accounts for 7,057; the remainder is buffer and inverter insertion from the same re-optimization and is not integrity logic.

Comparison against a measured conventional ECC checker. To make the comparison measured on both sides, we implemented and synthesized a conventional per-block ECC checker under the same tool, library, and clock constraint: the same $(126, 112, t{=}2)$ code as Design A over the same 112-bit payload, differing only in that the 14 parity bits are read from a separate SRAM port rather than from bit 0 of the weights, and without whitening. A detect-only and a correcting variant were built, so that the cost of the separate parity port is isolated from the cost of correction (Table 22). The correcting variant was verified over 10,000 random blocks at each fault level: zero, one, and two flips are corrected exactly in every case, and the miscorrection rate above capacity is measured. Design A costs 671 μm^2 more logic than the conventional detect-only checker, about 13%, and in exchange eliminates the parity array entirely; it is also faster, 1.27 ns against 2.51 ns, because reading parity from the weight stream avoids the separate SRAM port on the critical path. Against the correcting variant, Design A is 40% smaller and $2.6\times$ faster, at the cost of moving recovery

Table 21: Complete measured overhead (Synopsys DC V-2023.12-SP5, gscl45nm, 100 MHz constraint). Block A compares the full accelerator with and without a checker; block B characterizes the checkers themselves.

Block A. Full accelerator			
	Unprotected	With Design A	With Design B
Full-top critical path (ns)	30.38	31.14	30.99
Dynamic power (mW)	113.11	113.81	113.75
Leakage (mW)	7.581	7.586	7.599
Weight memory added (bytes)	0	0	0
Stall cycles added	0	0	0
Block B. Checker alone			
		Design A	Design B
Area (μm^2)		5,763.94	7,084.55
Area, clock-gated (μm^2)		5,824.95	7,173.72
Area overhead over the accelerator (%)		0.4118	0.5071
Critical path (ns)		1.27	1.31
Power (mW)		0.797	0.619
Single-flip detection, bits 7 / 6 / 5 / 0 (%)		100 / 100 / 100 / 100	0 / 100 / 50.4 / 1.4

into software. The baselines protect the same 112-bit payload as Design A so that logic areas are comparable; a conventional design covering all eight bits would use a $(142, 128, t{=}2)$ code with the same 14 parity bits and a correspondingly larger gather.

Table 22: Design A against a measured conventional per-block ECC checker (same code and payload, same tool, library, and clock; Synopsys DC V-2023.12-SP5, gscl45nm, 100 MHz constraint).

	Design A (ours)	**Conv. detect-only**	**Conv. correcting**
Parity location	bit 0 of the weights	separate SRAM	separate SRAM
Checker area (μm^2)	5,763.94	5,092.84	9,646.46
Critical path (ns)	**1.27**	2.51	3.28
Parity SRAM, CIFAR-10 CNN	**0 B**	15,355 B	15,355 B
Parity SRAM, ResNet-18	**0 B**	1.22 MB	1.22 MB
Correction buffer	0	0	16 B
Miscorrection at $t{+}1$	flags	flags	47.21%

Coexistence with the software recovery code. The hardware checker never reads the software parity, but the two must share the weight bits, so we instantiated the software code on the same platform and measured the interaction. For this small CNN we use the light code of Section 3.3, BCH$(127, 99, t{=}4)$ over GF(2^7) with primitive polynomial $x^7 + x + 1$ (generator degree 28), with one parity bit per weight placed in bit 1, a simplified placement in place of the cross-layer STDM carrier of Section 3.4. The 140,384 weights form 1,106 codewords (1,105 full plus one shortened tail of 49 weights) carrying 30,968 parity bits; embedding nudges 15,565 weights (11.09%) by ± 2. The pipeline embeds the software parity first, then Design A computes its bit-0 parity over the resulting payload, so both codes decode at zero BER. Design A reads bit 1 as opaque payload; it holds no $x^7 + x + 1$ generator and no $t = 4$ locator, so the two codes share weight bits without sharing interpretation. The software code was verified exhaustively over all 8,129 patterns of zero, one, and two errors and over 100,000 random three-error and 100,000 random four-error patterns, all corrected exactly; at five errors, 96.91% of patterns are flagged and 3.09% miscorrected. On the same 100 images the accelerator with Design A alone scores 88 and with both codes 87 (unprotected 86), the one-image difference being the ± 2 perturbation of the second embedding; no false flag is raised in either configuration. An in-line hardware decoder for this code would need four syndromes, a four-error locator, and a larger Chien search, several thousand gates above the 5,764 μm^2 detect tap, which is why recovery stays in software.

Storage and fetch traffic. The protected weight artifact is byte-identical to the unprotected one. Table 23 sets three columns side by side for eleven models from 3.5M to 8B weights: the unprotected baseline, conventional per-block ECC with its parity in a separate array fetched alongside the weights on every inference, and SANAI. The comparison is of storage and traffic at block-level integrity, since conventional ECC corrects in hardware whereas SANAI detects in hardware and recovers

in software from the embedded parity. SANAI matches the baseline in every column because it adds no stored bit; conventional ECC adds 10.94% storage and 9.86% fetch traffic regardless of model size, so the absolute cost scales linearly with the weight count, from 383 KB and 61 μJ per inference on MobileNet-V2 to 878 MB and 141 mJ on Llama-3-8B-Instruct at the per-bit DRAM energy of Han et al. (2016) and Horowitz (2014) (640 pJ per 32-bit access, 20 pJ per bit); on the CIFAR-10 CNN of this platform the saving is 2.46 μJ. These energies are modeled under one fetch per weight per inference: the platform holds weights in on-chip ROM and no DRAM access was measured. A further consequence of zero storage overhead is that conventional ECC can displace a model from on-chip memory that SANAI leaves resident: for a model whose unprotected size falls in the top 9.86% of an on-chip weight memory, the conventional 10.94% inflation pushes it off chip. At the same per-access energies (0.15625 pJ/bit SRAM against 20 pJ/bit DRAM, a factor of 128) the weight-fetch energy then differs by $142\times$, 1.311 μJ against 186.12 μJ per inference for a model sized at a 1 MB on-chip weight memory. This too is analytical, derived from measured storage sizes and published per-access energies.

Table 23: Storage and modeled weight-fetch energy per inference: unprotected baseline versus conventional per-block ECC (14 parity bits per 16 INT8 weights in a separate array) versus SANAI (parity inside the weights), for eleven models. Energy assumes one fetch per INT8 weight per inference from DRAM at 20 pJ per bit (640 pJ per 32-bit access, Table I of Han et al. (2016), after Horowitz (2014)); modeled, not measured. SANAI equals the baseline in every column; conventional ECC adds 10.94% storage and 9.86% traffic.

		Stored weights + parity			Weight-fetch energy per inference (modeled)			
Model	**Weights**	**Baseline**	**Conv. ECC**	**SANAI**	**Baseline**	**Conv. ECC**	**SANAI**	**Saving vs conv.**
MobileNet-V2	3.5M	3.5 MB	3.9 MB	**3.5 MB**	560 μJ	621 μJ	**560 μJ**	61 μJ
EfficientNet-B0	5.3M	5.3 MB	5.9 MB	**5.3 MB**	848 μJ	941 μJ	**848 μJ**	93 μJ
ResNet-18	11.7M	11.7 MB	13.0 MB	**11.7 MB**	1.9 mJ	2.1 mJ	**1.9 mJ**	205 μJ
ResNet-50	25.6M	25.6 MB	28.4 MB	**25.6 MB**	4.1 mJ	4.5 mJ	**4.1 mJ**	448 μJ
Swin-T	28.2M	28.2 MB	31.3 MB	**28.2 MB**	4.5 mJ	5.0 mJ	**4.5 mJ**	494 μJ
AlexNet	61.1M	61.1 MB	67.8 MB	**61.1 MB**	9.8 mJ	10.8 mJ	**9.8 mJ**	1.1 mJ
ViT-B/16	85.8M	85.8 MB	95.2 MB	**85.8 MB**	13.7 mJ	15.2 mJ	**13.7 mJ**	1.5 mJ
VGG-16	138.4M	138.4 MB	153.5 MB	**138.4 MB**	22.1 mJ	24.6 mJ	**22.1 mJ**	2.4 mJ
Qwen2-0.5B	494M	494 MB	548 MB	**494 MB**	79.0 mJ	87.7 mJ	**79.0 mJ**	8.6 mJ
Llama-3.2-1B	1.24B	1.24 GB	1.38 GB	**1.24 GB**	198 mJ	220 mJ	**198 mJ**	21.7 mJ
Llama-3-8B-Instruct	8.03B	8.03 GB	8.91 GB	**8.03 GB**	1.28 J	1.43 J	**1.28 J**	141 mJ

Per weight: baseline and SANAI store 8 bits and fetch 8 bits; conventional ECC stores and fetches 8.875 bits ($+10.94\%$ storage, $+9.86\%$ traffic).

Sensitivity. Checker area and energy per verified block grow with block length L, since the gather and its per-position store scale with the block; correction capacity t contributes comparatively little, as the syndrome engine is a small fraction of the checker. Measured at the synthesized operating point ($L = 16, t = 2$) and modeled elsewhere, area overhead ranges from 0.33% at $L = 8$ to 1.4% at $L = 64$, crossing 0.5% between $L = 16$ and $L = 32$.

Scope and limitations. Design B's sign-bit blind spot is structural to the power-of-two lattice; Design A does not share it, which is the principal reason it is the primary design. The software code shares weight bits with Design A's payload: the hardware never decodes it, but fully disjoint placement is not possible because Design A reads every bit above the LSB. The checker is integrated on one output-stationary array with a workload sized to its on-chip ROM; the checker logic is unchanged for larger models, but their end-to-end energy is modeled from the weight count (Table 23) rather than simulated, and all DRAM-related energy is modeled from published per-access constants. The library provides no Verilog cell models, so power is annotated from RTL-level switching activity rather than gate-level simulation with back-annotated delays.

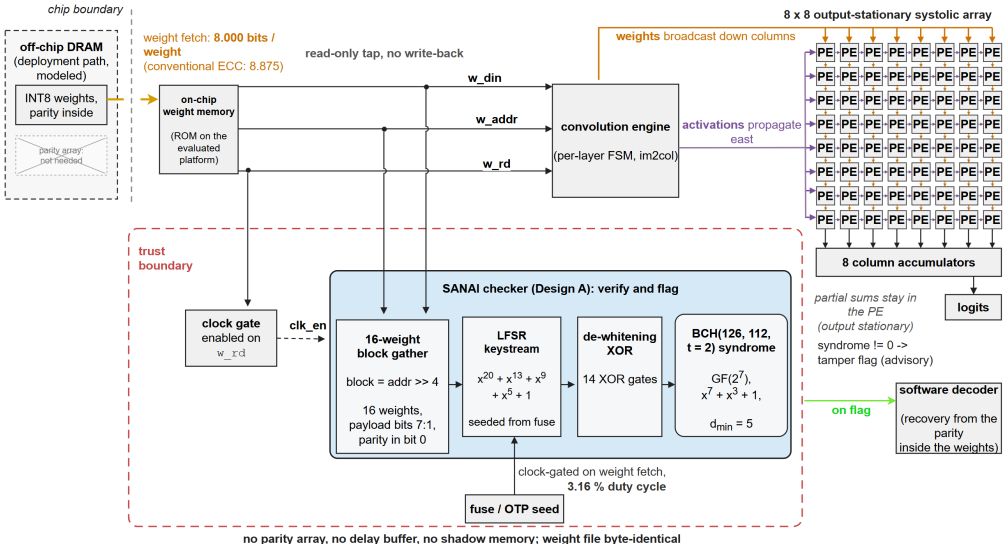


Figure 14: SANAI Design A checker in the 8×8 output-stationary accelerator: a read-only tap on `w_rd`, `w_addr`, and `w_din` between weight memory and the convolution engine, with `w_rd` gating the clock and the other two feeding the 16-weight block gather, LFSR keystream, de-whitening XOR, and BCH$(126, 112, t{=}2)$ syndrome. A nonzero syndrome raises an advisory tamper flag to the software decoder; there is no write-back, buffer, or parity array, and the weight file is byte-identical. The trust boundary encloses checker, clock gate, and seed.