Journal of the Brazilian Computer Society, 2026, 32:1, doi: 10.5753/jbcs.2026.7465
This work is licensed under a Creative Commons Attribution 4.0 International License.

MLBFA-Pruned: Optimizing Bit-Flip Attacks on Quantized DNNs via ML-Guided Ranking

José Ronaldo [**Laboratório de Sistemas Digitais (LSD), Federal Institute of Ceará** | ***ronaldoferrreira.b@gmail.com***]
Otávio Alcântara de Lima [**Laboratório de Sistemas Digitais (LSD), Federal Institute of Ceará** | ***otavio@ifce.edu.br***]

Laboratório de Sistemas Digitais (LSD), Federal Institute of Ceará, Maracanaú, CE, Brazil.

Received: 26 January 2026 • **Accepted:** 09 October 2026 • **Published:** 17 November 2026

Abstract. Quantized deep neural networks (DNNs) remain vulnerable to bit flips in stored weights, motivating compact attacks for safety-critical and resource-constrained systems. This work extends MLBFA, which combines vulnerability-model (VM) guided Top-N candidate filtering with heuristic search. The proposed MLBFA-Pruned applies backward elimination to successful solutions, removing individually redundant flips while preserving the target-accuracy condition. We also evaluate regression-based bit-specific ranking trained with additional statistical fault injection paradigms and expand the protocol analysis. Backward elimination reduced every campaign-best Greedy Bit Search solution for which paired before-and-after data were available, including the 4-bit ResNet-20 solution from 15 to 11 flips. Classification-based ranking produced smaller selected best attacks than regression-based ranking for three of four architectures. Because the MLBFA-Standard and MLBFA-Pruned campaigns were executed independently under potentially different settings, their comparison is descriptive. The primary descriptive comparison with Rakin et al. uses successful-attack medians for both methods, restricts MLBFA-Pruned to the balanced $K = 2{,}024$ stratum, and reports success fractions and dispersion. MLBFA-Pruned yielded a lower median in 10 of 12 architecture–precision settings. A complementary exploratory minimum-to-minimum comparison yielded lower MLBFA-Pruned minima in eight settings, equal minima in two, and higher minima in two. Overall, VM-guided search with post-search elimination produces compact successful attacks, with measured reductions applying specifically to solution cardinality.

Keywords: Deep Neural Networks, Bit-Flip Attack, Fault Injection, Machine Learning, Vulnerability Model, Reliability.

1 Introduction

Deep neural networks (DNNs) are increasingly used in systems whose failures may have safety or security consequences. Their parameters can be corrupted by aging, radiation-induced upsets, or deliberate physical attacks [Lin *et al*., 2025; Global Desk, 2025]. Bit-flip attacks (BFAs) are especially concerning because a small number of changes to stored weights may silently compromise inference [Qian *et al*., 2023]. RowHammer, for example, can induce such changes by repeatedly accessing adjacent memory rows [Kim *et al*., 2014; Breier and Hou, 2022]. Evaluating which faults are harmful is therefore an important part of DNN reliability assessment [Bosio *et al*., 2019; Ruospo *et al*., 2020; Ahmadilivani *et al*., 2024; Liu *et al*., 2017; Sabbagh *et al*., 2019; Qian *et al*., 2023; Torres-Huitzil and Girau, 2017; Yu *et al*., 2025].

Software-based fault injection (FI) provides a controllable and comparatively inexpensive way to perform this evaluation [Arlat *et al*., 1990; Ruospo *et al*., 2021, 2023a]. Existing frameworks can emulate faults and measure their effects [Mahmoud *et al*., 2020; Chen *et al*., 2020; Huang *et al*., 2024; Govarini *et al*., 2023; Zheng *et al*., 2021; Agarwal *et al*., 2022; Reagen *et al*., 2018; Göğebakan *et al*., 2024], but exhaustively testing every bit of a modern network is generally infeasible [Syed *et al*., 2023; Ruospo *et al*., 2023b; Govarini *et al*., 2023; Zhang *et al*., 2022, 2023]. Statistical FI and machine-learning vulnerability models address this scale by learning from sampled injections, although they have primarily been used to estimate reliability or guide selective protection [Ruospo *et al*., 2023b, 2025; Leveugle *et al*., 2009; Zhang *et al*., 2022, 2023; Traiola *et al*., 2023b,a; Huang *et al*., 2020]. Conversely, BFA methods commonly combine a single ranking signal, such as a gradient or saliency score, with a heuristic search such as Progressive Bit Search (PBS) [Rakin *et al*., 2019, 2022; Hector *et al*., 2022; Dong *et al*., 2023]. This leaves an opportunity to use multidimensional vulnerability prediction to focus attack search on a small set of promising, explicitly indexed bits.

We study quantized DNNs because they are both operationally important and nontrivial attack targets. Low-precision integer representations reduce memory, computation, and energy costs in TinyML or edge systems and their fixed-width codes also give each stored bit a direct and bounded numerical meaning, unlike floating-point exponent faults that can change a value by orders of magnitude. This bounded representation does not make significant bits harmless, but it makes arbitrary flips relatively ineffective: Rakin et al. [Rakin *et al*., 2019] report little accuracy loss from 100 random flips in an 8-bit ResNet-18, while 13 strategically selected flips reduce the same model to near-chance accuracy. Finding a small successful set is consequently a combinatorial search problem. Our primary objective is the minimum-cardinality attack: find the fewest flips that reduce accuracy below a predefined near-chance threshold. This objective reflects a constrained attacker for whom every additional physical fault increases effort and exposure; it differs

from maximizing damage under a fixed flip budget, which we formulate separately in Section 3.

MLBFA, our machine-learning-guided BFA framework, was introduced in previous work [Ronaldo *et al*., 2025] to connect vulnerability modeling with BFA search. The original pipeline learns a classification-based vulnerability model (VM) from a limited statistical fault injection (SFI) campaign, ranks the admissible faults using a multidimensional representation, retains a Top-N candidate pool, and searches that pool with Greedy, Evolutionary, or Progressive Bit Search. Section 3 presents the complete pipeline and the extensions developed in this article, including the feature definitions, fault representation, ranking procedures, and search objectives.

This article extends that previous work by preserving the original classification-based MLBFA pipeline while broadening both the method and its evaluation. Its main additional contributions are:

1. it adds *post-search backward elimination*, which removes individually redundant flips from a successful supplied solution and defines the MLBFA-Pruned operator;
2. it investigates regression-based, bit-specific vulnerability ranking using training data generated by additional Data-Aware and iterative SFI paradigms at bit, layer, and network granularities; and
3. it expands the experimental analysis of the framework, including attack outcomes, protocol differences, paired before–after results for backward elimination, a primary balanced median-based comparison, and a complementary exploratory minimum-to-minimum comparison with Rakin et al.

The framework uses two complementary reduction mechanisms at different stages. VM-guided Top-N filtering restricts the candidate pool before search and is part of both variants. MLBFA-Standard returns the solution produced by the search, whereas MLBFA-Pruned subsequently applies backward elimination to remove redundant flips while preserving attack success. Because backward elimination operates on a supplied solution, its effect is evaluated through paired before–after records; comparisons between independently executed Standard and Pruned campaigns are treated separately as comparisons of final outcomes.

Our numerical reference is Rakin et al. [Rakin *et al*., 2019], selected for its untargeted near-chance objective and its published CIFAR-10 results for the same four architecture depths and three quantization levels, rather than as a claim that it is the latest or universally strongest BFA. Its gradient-guided PBS contrasts with MLBFA's learned multidimensional ranking and optional post-search backward elimination. Subsequent work addresses complementary settings: T-BFA [Rakin *et al*., 2022] directs selected inputs toward a target class; Hector et al. [Hector *et al*., 2022] examine evaluation variability, architecture dependence, and adversarial budgets; and Dong et al. [Dong *et al*., 2023] study training-assisted attacks in which the adversary helps construct a model that is later compromised by a small number of flips. These objectives and attacker capabilities differ from our untargeted attack on an already trained checkpoint. We acknowledge these later contributions without treating their reported flip counts as interchangeable baselines. Our cross-study comparison is descriptive because search procedures, configurations, budgets, and stopping rules are not matched; no superiority over all later BFA methods is established.

The remainder of this paper is organized as follows. Section 2 presents the SFI foundations used to construct the VM training data. Section 3 defines the MLBFA pipeline, attack objectives, and the boundary between its two variants. Section 4 describes the experimental protocols and reports the results and consistency analyses. Section 5 concludes the paper and outlines future work.

2 Statistical Fault Injection Foundations

This section introduces only the SFI foundations used to construct the VM training data; the attack model and search procedures are defined in Section 3. SFI models fault injection as random sampling without replacement from a finite population of N candidate faults [Leveugle *et al*., 2009]. An injection is considered a success when the selected fault produces the predefined failure condition; hence, $p \in [0, 1]$ denotes the unknown population failure rate, i.e., the probability that a uniformly selected candidate fault produces that condition. To estimate p with a margin of error ε and confidence level $1 - \alpha$, the required sample size n is derived using the finite population correction (FPC) [Leveugle *et al*., 2009]:

$$n = \frac{N\, t_{1-\alpha/2}^2\, p(1-p)}{\varepsilon^2(N-1) + t_{1-\alpha/2}^2\, p(1-p)}, \tag{1}$$

where $t_{1-\alpha/2}$ is the standard normal quantile (e.g., 2.576 for 99% confidence). We evaluate paradigms classified into the following categories:

2.1 Data-Unaware One-Step SFI

The *Data-Unaware* (or Conservative) paradigm determines the sample size n by fixing the probability $p = 0.5$ in Eq. (1) and applying it at the specific granularity of the analysis, assuming that every injected fault has the same probability of success or failure [Ruospo *et al*., 2023b]. This choice is conservative because $p(1-p) = 1/4 - (p-1/2)^2$ is maximized at $p = 0.5$; since the sample size in Eq. (1) increases monotonically with $p(1-p)$, this value yields the largest required sample for fixed N, ε, and α, thereby guaranteeing the target margin of error without prior knowledge of the actual failure rate.

2.2 Data-Aware One-Step SFI

Targeting 8-bit quantized weights, the *BIT-ONE* paradigm exploits the intuition that bit-flips in different positions cause varying impact magnitude. Inspired by [Ruospo *et al*., 2023b], we define a weighted average distance $D_{\text{avg}}(i, \ell)$ for bit-plane i in layer ℓ, which captures the structural impact of a fault. Let $a = 0$ and $b = 0.5$ denote the lower and upper endpoints of the proxy-probability interval, respectively, and let $D_{\min}$ and

$D_{\max}$ denote the minimum and maximum values of $D_{\text{avg}}(i,\ell)$ over all bit-plane–layer pairs considered in the campaign. This metric is mapped to a proxy probability $p(i,\ell) \in [a,b]$ via min-max normalization:

$$p(i,\ell) = a + \frac{(D_{\text{avg}}(i,\ell) - D_{\min})(b-a)}{D_{\max} - D_{\min}}. \quad (2)$$

Bits causing larger perturbations receive $p(i,\ell)$ closer to 0.5, increasing the sample size via Eq. (1). The campaign samples $n(i,\ell)$ faults independently per bit-plane in a single step.

2.3 Iterative SFI Paradigms

Iterative campaigns refine the sample size n_k dynamically based on the observed failure rate $\hat{p}_k$ after k iterations [Ruospo *et al.*, 2025]. To accelerate convergence, we employ the "E–P curve" methodology, which adjusts the target margin $\varepsilon_{\text{next}}$ based on $\hat{p}_k$ using a quadratic function $\varepsilon_{\text{poly}}(\hat{p})$ clipped to $\varepsilon_{\text{goal}}$ and the current margin $\hat{\varepsilon}_k$:

$$\varepsilon_{\text{next}} = \max\left\{\varepsilon_{\text{goal}},\ \min\left\{\varepsilon_{\text{poly}}(\hat{p}_k),\ \frac{\hat{\varepsilon}_k}{3}\right\}\right\}. \quad (3)$$

The updated sample size n_{k+1} is calculated by substituting $p = \hat{p}_k$ and $\varepsilon = \varepsilon_{\text{next}}$ into Eq. (1). We implement this mechanism across three granularities:

- *BIT-ITER*: Applies the iterative loop to each bit-plane i independently. This corresponds to *Bit-wise SFI*, where the percentage of critical faults is estimated for every bit position.
- *LAYER*: Treats each network layer as a distinct population. This aligns with *Layer-wise SFI*, where the convergence is enforced strictly per layer to estimate the criticality of specific architectural components.
- *NET*: Treats the entire network as a single global population of faults. This corresponds to *Network-wise SFI*, offering the fastest convergence for estimating the total percentage of critical faults affecting the entire convolutional neural network model.

3 Proposed Framework

MLBFA, introduced in the conference version [Ronaldo *et al.*, 2025], is designed to compromise a DNN with a minimal number of bit-flips. As delimited in the Introduction, the present article retains the classification-based conference pipeline as MLBFA-Standard and adds backward elimination, regression-based bit-specific ranking with additional SFI paradigms, and expanded paired, outcome, and protocol analyses. The framework, illustrated in Figure 1, separates three inherited operations: (1) building a VM to identify susceptible DNN weights; (2) ranking the complete admissible fault universe and applying VM-guided Top-N candidate filtering before search; and (3) searching within the resulting candidate pool for a solution that degrades accuracy to the near-chance state defined below. The extension presented in this work adds a fourth, post-search operation in MLBFA-Pruned: backward elimination of redundant flips from a successful solution. The search process uses the classic PBS [Rakin *et al.*, 2019, 2022], Greedy Bit Search (GBS), and Evolutionary Bit Search (EBS). The attack is untargeted and white-box, assuming full access to the DNN's parameters [Qian *et al.*, 2023]. We focus on 4-, 6-, and 8-bit quantized DNNs because their practical embedded use and explicit storage-level bit semantics make the search for small, interacting sets of critical flips both operationally relevant and methodologically nontrivial.

Stage I: VM Construction The core of our framework is the VM, a machine learning model designed to predict fault susceptibility without the prohibitive cost of exhaustive injection [Traiola *et al.*, 2023a; Govarini *et al.*, 2023; Ruospo *et al.*, 2023a]. The inherited classifier uses the conference paper's layer-wise Data-Unaware SFI data. The extension presented in this work additionally uses the BIT-ONE, BIT-ITER, LAYER, and NET paradigms defined in Section 2 to train regression VMs and evaluate how the SFI granularity is associated with the resulting ranking.

For each SFI paradigm, the sample size n is determined by either Eq. (1) or Eq. (2), ensuring statistical significance with a 99% confidence level. Each injected fault is evaluated on an internal subset of K images sampled from the CIFAR-10 test set, where $K \in \{1{,}024, 2{,}024, 10{,}000\}$ and the $K = 10{,}000$ condition uses the complete test set. These are internal test subsets, not a separate validation partition. The resulting accuracy change maps each perturbed parameter to its observed impact. From these data, we train two types of VMs: (i) an Extra-Trees **classifier**, trained on data generated by the *Data-Unaware* paradigm at *layer-wise* granularity (Eq. (1) with $p = 0.5$), which predicts a binary vulnerability label based on an accuracy-drop threshold; and (ii) a Random-Forest **regressor**, trained on data from *Data-Aware* (*BIT-ONE*) and *Iterative* (*BIT-ITER*, *LAYER*, *NET*) paradigms, which predicts the continuous accuracy drop.

An initial exploratory pool of 21 candidate features was extracted for each target weight. A predefined screening step excluded `sparsity`, the raw `E_in` and `E_out` energies, and the raw saliencies derived from Single-shot Network Pruning (SNIP) and SynFlow, retaining the 16 features defined in Tables 1 and 2. For a convolutional layer, let $w_{o,i,u,v}$ denote a weight; for a linear layer, the spatial indices are omitted. We define $g_t(w) = \partial \mathcal{L}_t / \partial w$ as the gradient computed on minibatch t, and let L denote the number of eligible Conv2d and Linear layers. Layer- and channel-level quantities are broadcast to all weights to which they apply. The values are extracted as defined below, without dataset-wide scaling at this stage. Monospaced names match the implemented extractor fields; the descriptive label "root-mean-square (RMS) gradient" is used for the field `hii` to avoid implying that a Hessian diagonal was explicitly computed.

Data-dependent predictors are extracted from CIFAR-10 using channel-wise normalization with means $(0.4914, 0.4822, 0.4465)$ and standard deviations $(0.247, 0.243, 0.261)$, a minibatch size of 128, and random seed 0. The training split is used to compute `gradient`, `hii`, and the SNIP-derived score, whereas the test split is used only for `influence_log`. The SynFlow-derived score and the structural predictors are data-free. The predictor set builds on the vulnerability-modeling features of Zhang et al. [Zhang *et al.*, 2022, 2023] and incorporates saliency predictors

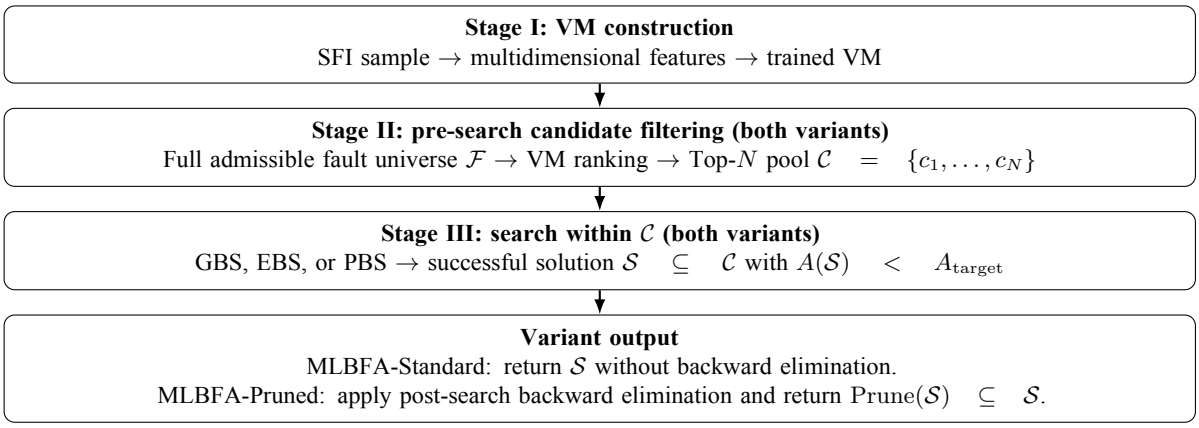

Figure 1. MLBFA pipeline and variant boundary. VM-guided Top-N filtering restricts the candidate pool before search and is common to MLBFA-Standard and MLBFA-Pruned. Only MLBFA-Pruned applies post-search backward elimination to reduce the cardinality of an already successful solution.

adapted from SNIP [Lee *et al*., 2019] and SynFlow [Tanaka *et al*., 2020].

The fixed screening above is the feature-selection step; it is not learned from the data. Both 8-bit training pipelines additionally retain `bit_index`; it identifies the bit-specific regression candidate, whereas classification-based ranking later fixes it to the most significant bit (MSB). For iterative BIT- and LAYER-wise datasets, the pipelines also retain one numeric pair identifier. Consequently, the design matrices contain the 16 defined features plus one or two additional columns, for totals of 17 or 18 columns, respectively. All numeric columns are median-imputed and standardized before fitting either the Extra-Trees classifier or the Random-Forest regressor.

Stage II: Vulnerability Ranking and Pre-Search Candidate Filtering Using the fitted preprocessing pipeline and the corresponding trained VM, we assign a vulnerability score to each admissible bit-flip candidate. Let $q \in \{4, 6, 8\}$ be the quantization bit-width, let ℓ identify a layer, let j be the flattened index of a weight within that layer, and number the bits by $b \in \{0, \ldots, q-1\}$, from the least significant bit (LSB) to the MSB. Thus, one candidate is the concrete bit flip $c = (\ell, j, b)$, and $\mathcal{F}$ denotes the complete set of admissible candidates under the corresponding fault model. In the classification-based approach, the classifier assigns the weight at (ℓ, j) the score $\hat{p}_{\ell,j} = \Pr(\text{Vulnerable} \mid \mathbf{x}_{\ell,j})$, and this score is attached to the fixed-MSB candidate $c = (\ell, j, q-1)$. In the regression-based approach, the score $\hat{y}_{\ell,j,b}$ estimates the accuracy-drop magnitude of a bit-specific candidate, so b may vary across candidates. We sort the elements of $\mathcal{F}$ in descending score order and perform VM-guided Top-N candidate filtering, producing

$$\mathcal{C} = \{c_1, \ldots, c_N\} \subseteq \mathcal{F}.$$

This pre-search operation is the mechanism that reduces the candidate space presented to Stage III. It is common to MLBFA-Standard and MLBFA-Pruned and requires no additional fault injections.

Stage III: Search and Optional Post-Search Solution Refinement The final stage uses the vulnerability ranking produced in Stage II to search for a compact set of bit flips. Let $\mathcal{C} = (c_1, \ldots, c_N)$ denote the ordered pool of the Top-N candidates, with

$$c_k = (\ell_k, j_k, b_k),$$

where ℓ_k is the layer, j_k is the flattened within-layer weight index, and b_k is the specific bit position to be toggled. For classification, $b_k = q - 1$ for every candidate (the MSB); for regression, b_k is candidate-specific and may take any admissible bit position. Consequently, two tuples that refer to different bits are distinct candidates even if they refer to the same weight. Let $\mathcal{S} \subseteq \mathcal{C}$ denote a set of these concrete bit flips selected for simultaneous injection. For an internal subset $\mathcal{D}^{(K)}_{\text{subtest}} \subseteq \mathcal{D}_{\text{test}}$, we distinguish

$$A^{(K)}_{\text{subtest}}(\mathcal{S}) = \text{Acc}(\mathcal{S}; \mathcal{D}^{(K)}_{\text{subtest}}),$$
$$A_{\text{test}}(\mathcal{S}) = \text{Acc}(\mathcal{S}; \mathcal{D}_{\text{test}}).$$

The former guides internal search evaluations, whereas the latter is the operational final-evaluation metric used to confirm attack success and post-search removals. In the classification experiments, $\mathcal{D}_{\text{test}}$ is the complete 10,000-image CIFAR-10 test set. The regression campaign instead uses the stored 1,024-sample evaluation set specified in Section 4.6; its reported endpoints should not be interpreted as independently verified 10,000-image accuracies. Throughout the remainder of the paper, an unqualified $A(\mathcal{S})$ denotes $A_{\text{test}}(\mathcal{S})$ under the stated protocol, and $A(\emptyset)$ denotes the corresponding fault-free baseline. For a balanced C-class problem, we define the near-chance failure threshold as

$$A_{\text{target}} = A_{\text{chance}} + \delta = \frac{1}{C} + \delta.$$

For CIFAR-10, $C = 10$ and hence $A_{\text{chance}} = 10\%$. Following the evaluation protocol of Rakin et al. [Rakin *et al*., 2019], we set $\delta = 0.01$, giving $A_{\text{target}} = 11\%$, and confirm an attack only if $A_{\text{test}}(\mathcal{S}) < A_{\text{target}}$. Because test-set images are queried during search and post-search processing, A_{test} is treated as an operational attack metric rather than a strictly held-out estimate of generalization.

The primary attack objective can therefore be stated as the constrained minimum-cardinality problem

$$\min_{\mathcal{S} \subseteq \mathcal{C}} |\mathcal{S}| \quad \text{subject to} \quad A(\mathcal{S}) < A_{\text{target}}. \tag{4}$$

Table 1. Operational definitions of the 16 features retained for model fitting (Part I: intrinsic, architectural, and gradient-based features).

Predictor	Operational definition and transformation	Scope and input	Interpretation and source
`abs_value`	$\lvert w\rvert$; no additional scaling	weight; checkpoint	Weight-magnitude signal adapted from Zhang *et al*. [2022, 2023].
`sign`	$\mathrm{sign}(w)$, with zero encoded as $+1$	weight; checkpoint	Retains polarity under signed quantization; defined in this work.
`depth_norm`	$d(\ell) = r(\ell)/(L-1)$ for ordered eligible-layer rank r	layer; checkpoint	Normalized input-to-output location, adapted from Zhang *et al*. [2022, 2023].
`param_index`	Zero-based row-major index of w within its layer tensor; no scaling	weight; checkpoint	Implementation-defined ordinal coordinate used in this work.
`stride`	First spatial stride of Conv2d; 1 for Linear	layer; checkpoint	Spatial-downsampling descriptor defined in this work.
`dilate`	First spatial dilation of Conv2d; 1 for Linear	layer; checkpoint	Kernel-spacing descriptor defined in this work.
`groups`	Number of convolution groups; 1 for Linear	layer; checkpoint	Channel-connectivity descriptor defined in this work.
`gradient`	$\bar{g}(w) = B_g^{-1} \sum_{t=1}^{B_g} g_t(w)$; signed, unscaled mean	weight; CIFAR-10 train	First-order loss sensitivity adapted from Zhang *et al*. [2022, 2023].
`calc_times`	$H_{\mathrm{out}} W_{\mathrm{out}}$ from a $1\times3\times32\times32$ dummy forward pass; 1 for Linear	layer; dummy input	Nominal output-position reuse count, adapted from Zhang *et al*. [2022, 2023]; it is not a count of nonzero border multiplications.
Root-mean-square (RMS) gradient (`hii`)	$h(w) = \sqrt{B_h^{-1} \sum_{t=1}^{B_h} g_t(w)^2}$; no additional scaling	weight; CIFAR-10 train	Square root of a diagonal empirical-Fisher estimate; a gradient-curvature proxy, not an explicitly computed Hessian diagonal.

Note. $B_g = 50$ and $B_h = 4$ are the maximum numbers of training minibatches used by the implemented extractor. "Checkpoint" denotes a data-free extraction from the trained checkpoint and architecture.

Severe degradation is encoded as the feasibility constraint. In CIFAR-10, a feasible solution must reduce the model below the 11% near-chance threshold, i.e., to practically random-guess behavior. Once this operational failure condition has been met, further reducing accuracy does not change the success status, whereas each additional controlled flip entails another physical memory manipulation, increases operational effort and the requirements imposed on the fault mechanism, and enlarges the attack footprint and its opportunity for detection. Cardinality is therefore an operational proxy for attack cost under the stated threat model, while physical differences among bit positions remain outside this proxy. Restricted to the Top-N pool and finite computational budgets, MLBFA heuristically seeks small feasible sets within the evaluated search domain.

Maximizing damage under a fixed flip budget is a complementary formulation:

$$\min_{\substack{\mathcal{S}\subseteq\mathcal{C} \\ |\mathcal{S}|\leq B}} A(\mathcal{S}). \tag{5}$$

This formulation asks for the lowest residual accuracy attainable with at most B flips, whereas Eq. (4) asks for the least costly attack that already reaches the predefined failure state. The two objectives need not select the same set. The accuracy-based fitness used internally by EBS under $B_{\max}$ is aligned with the damage-oriented formulation in Eq. (5); the final success test and, in MLBFA-Pruned, backward elimination address the article's primary cardinality objective.

Finally, $B_{\max}$ denotes the algorithm- and version-specific maximum cardinality for runs that impose an explicit flip cap, so their feasible sets satisfy $|\mathcal{S}| \leq B_{\max}$. As illustrated in Figure 1, restricting the search to $\mathcal{C}$ reduces the combinatorial space before any search algorithm runs. GBS and EBS operate on explicitly indexed bits, rather than on abstract weights; PBS is included where specified in the experiments. GBS constructs a solution by evaluating the remaining one-candidate extensions and selecting the extension with the lowest internal-subset accuracy; its stopping rules depend on the executed classification or regression protocol, as detailed below and in Section 4.6. In contrast, EBS performs a stochastic, population-based search implemented with a genetic algorithm to uncover synergistic combinations of candidate bits that may be missed by purely sequential selection.

At the operator-definition level, MLBFA-Standard returns a supplied search solution, whereas MLBFA-Pruned applies the backward-elimination operator defined below to that same supplied solution. This operator-level distinction differs from the reported Standard and Pruned *campaigns*, which used different code versions and, in several cases, different search algorithms, budgets, internal subsets, and random-number consumption. Recorded before/after pairs isolate the effect of backward elimination. Candidate-pool reduction is attributable to Stage II, whereas a paired reduction produced by MLBFA-Pruned concerns final solution cardinality.

Greedy Bit Search. GBS implements a forward constructive strategy over $\mathcal{C}$. Starting from $\mathcal{S}_0 = \emptyset$, at step t the algorithm evaluates $\mathcal{S}_t \cup \{c\}$ on the internal subset for every remaining $c \in \mathcal{C} \setminus \mathcal{S}_t$ and appends the candidate yielding the lowest subset accuracy. The initial VM order therefore defines the candidate pool and acts as the tie order when strict accuracy comparisons are equal; GBS does not merely append the next VM-ranked item. Each selected prefix is then evaluated on the full test set. In the Standard and Pruned implementations evaluated here, crossing $A_{\text{target}} = 11\%$ does not stop this forward search: it continues to the configured cap unless candidates are exhausted; the optimized implementation also stops when accuracy reaches exactly zero. For the evaluated Pruned runs, backward elimination is invoked only after this capped search. The first successful prefix can be reported descriptively for Standard, but it is distinct from the full capped

Table 2. Operational definitions of the 16 features retained for model fitting (Part II: influence and transformed energy, sparsity, and saliency features).

Predictor	Operational definition and transformation	Scope and input	Interpretation and source
`influence_log`	$\log(1+I(w))$, where $I(w)=B_i^{-1}\sum_t \lvert\partial(\sum_k f_k(x_t))/\partial w\rvert$	weight; CIFAR-10 test	Log-compressed aggregate-logit sensitivity defined in this work; evaluation-split reuse is disclosed below.
`E_in_log`	$\log(1+\sum_{o,u,v} w_{o,i,u,v}^2)$; sum over o for Linear	input channel; checkpoint	Log-compressed source-channel weight energy, defined in this work.
`E_out_log`	$\log(1+\sum_{i,u,v} w_{o,i,u,v}^2)$; sum over i for Linear	output channel; checkpoint	Log-compressed destination-channel weight energy, defined in this work.
`s_logit`	$\log(\tilde{s}_\ell/(1-\tilde{s}_\ell))$, where $s_\ell=\lvert W_\ell\rvert^{-1}\sum_{w\in W_\ell}\mathbf{1}\{\lvert w\rvert<10^{-4}\}$ and $\tilde{s}_\ell=\mathrm{clip}(s_\ell,10^{-8},1-10^{-8})$	layer; checkpoint	Clipped-logit expansion of near-zero concentration, adapted from Zhang *et al.* [2022, 2023]; thresholds follow the implemented convention.
`snip_saliency_log`	$\log(1+S_\ell^{\mathrm{SNIP}})$, where S_ℓ^{SNIP} is the layer mean of $\lvert w\,\bar{g}_s(w)\rvert$ over B_s training minibatches	layer; CIFAR-10 train	Log-compressed SNIP-derived connection sensitivity [Lee *et al.*, 2019]; layer averaging and transformation are adaptations defined here.
`synflow_saliency_log`	$\log(1+S_\ell^{\mathrm{SF}})$, where S_ℓ^{SF} is the layer mean of $\lvert w\,\partial(\sum_k f_k(\mathbf{1};\lvert W\rvert))/\partial w\rvert$	layer; all-ones input	Log-compressed SynFlow-derived path-flow sensitivity [Tanaka *et al.*, 2020]; signed weights are restored after scoring.

Note. $B_s=10$ and $B_i=2$ are the maximum numbers of training and test minibatches, respectively, used by the implemented extractor. Layer-level SNIP- and SynFlow-derived values are broadcast to the corresponding weights.

solution supplied to post-search elimination.

Evolutionary Bit Search. To explicitly account for combinatorial interactions among faults, EBS formulates the search as a subset optimization problem solved via a genetic algorithm. A solution is encoded as a binary chromosome $\mathbf{z}=(z_1,\ldots,z_N)\in\{0,1\}^N$, where $z_k=1$ selects the specific flip $c_k=(\ell_k,j_k,b_k)$ (and hence toggles bit b_k of weight j_k in layer ℓ_k), whereas $z_k=0$ leaves that bit unchanged. Chromosomes are restricted to an architecture- and implementation-specific maximum flip budget $B_{\max}$, i.e., $\|\mathbf{z}\|_0\le B_{\max}$; any offspring violating this constraint is repaired by disabling excess active genes. Here, $B_{\max}$ is an operational search cap rather than a hardware-derived estimate of an attacker's physical capability. It both defines the largest attack admitted in a capped run and limits the combinatorial and evaluation cost of EBS. Its empirical calibration and the implementation-specific values are described in Table 3 and the accompanying text. The fitness of a chromosome is defined as the induced damage measured by the baseline accuracy drop:

$$F(\mathbf{z})=A(\emptyset)-A(\mathcal{S}(\mathbf{z})),\qquad \mathcal{S}(\mathbf{z})=\{c_k\in\mathcal{C}\mid z_k=1\}.$$

EBS evolves the population using tournament selection, single-point crossover, low-rate mutation, and elitism to preserve the best individuals across generations. To reduce evaluation overhead, populations are first scored on an internal subset. In both evaluated EBS implementations, only the best chromosome of a generation is evaluated on the full test set, and only when its subset fitness strictly exceeds the best subset fitness seen previously. Equal-fitness alternatives are therefore not fully evaluated. The search stops when such a fully evaluated chromosome reaches the target or when 70 generations are exhausted.

Post-Search Backward Elimination (GBS-Pruned and EBS-Pruned). Both GBS and EBS may return flip sets containing unnecessary candidates, either because the VM ranking includes false positives or because fault effects are non-additive, making some flips redundant once damaging combinations are formed. Therefore, whenever a successful solution $\mathcal{S}$ is found, we apply post-search backward elimination to reduce its cardinality. At each iteration t, we test the removal of each flip $f\in\mathcal{S}^{(t)}$ by evaluating the trial set $\mathcal{S}^{(t)}\setminus\{f\}$ and accepting a removal only if the attack remains successful, i.e., if $A(\mathcal{S}^{(t)}\setminus\{f\})<A_{\text{target}}$. Among all valid single-removal trials within the current iteration, the algorithm commits the removal that yields the *lowest* residual accuracy (strongest damage), updating $\mathcal{S}^{(t+1)}\leftarrow\arg\min_{\mathcal{S}^{(t)}\setminus\{f\}}A(\cdot)$. This backward refinement repeats until no further flip can be removed without violating the target constraint. Because every accepted update removes exactly one element and no update adds an element,

$$\mathrm{Prune}(\mathcal{S})\subseteq\mathcal{S},\qquad|\mathrm{Prune}(\mathcal{S})|\le|\mathcal{S}|.$$

In the executed implementation, candidates are enumerated in ascending VM score ("low-vulnerability-first"), but every one-flip removal is evaluated at each iteration and the valid trial with the lowest full-test accuracy is committed. The VM order therefore matters primarily as a tie-breaking order when trial accuracies are equal. Backward elimination may add network evaluations because it repeatedly tests single-flip removals. We therefore make no claim that this post-search procedure accelerates the preceding search or reduces the candidate space explored by that search.

Budget calibration and transfer to a new network. Before the reported campaigns, we performed exploratory attacks independently on each architecture. Starting from one admissible flip, we increased the allowed cardinality and searched for combinations that satisfied the same final criterion used in the main experiments, $A_{\text{test}}(\mathcal{S})<11\%$. The calibration stopped at the smallest explored cap for which the corresponding search implementation could reliably explore at least one successful ("lethal") combination under its fixed population/generation or greedy-search settings. Consequently,

Table 3. Maximum flip budgets used in the classification implementations. Values differ across architectures because they were calibrated independently rather than by a monotonic function of network depth. Regression-guided EBS uses the EBS-new limits; regression-guided GBS instead follows the Top-120 exhaustion rule described in Section 4.6.

Architecture	*EBS old*	*EBS new*	*GBS old*	*GBS new*
ResNet-20	9	9	15	15
ResNet-32	6	6	15	15
ResNet-44	12	12	22	22
ResNet-56	16	10	22	22

the values are architecture specific and need not increase with depth: they reflect the empirically observed difficulty of finding a successful combination in each network and search implementation, rather than parameter count alone. We also compared their order of magnitude with the flip counts reported by Rakin et al. [Rakin *et al*., 2019], who use the same below-11% objective. This literature comparison served as a calibration reference, while the resulting caps remain empirically selected and implementation specific.

For a previously unseen network, a user should therefore (i) fix the candidate-generation rule, search algorithm, validation subset, and success threshold; (ii) run a pilot sweep beginning at $B = 1$ and increase B in unit increments; and (iii) select the smallest B at which successful solutions are observed with the chosen search settings, repeating stochastic EBS trials so that a single favorable seed does not determine the cap. If the computational ceiling is reached without success, the outcome is recorded as failure within that ceiling. In the present experiments, $B_{\max}$ is a calibrated computational hyperparameter; its interpretation as an adversarial constraint is specific to the stated threat model. Table 3 reports both EBS versions because the later ResNet-56 implementation used a revised, version-specific cap (10 instead of 16).

4 Experimental Results and Analyses

This section first details the experimental setup and then reports the results by variant. All attack results before Section 4.6 use classification-based VM ranking unless explicitly stated otherwise. Section 4.2 evaluates the VM. Section 4.3 reports the classification-based MLBFA-Standard results, obtained with VM-guided search and no backward elimination. Section 4.4 reports the classification-based MLBFA-Pruned campaign and the paired backward-elimination analysis. Section 4.6 evaluates regression-based vulnerability ranking; within that section, the regression-guided EBS entries are executed under the MLBFA-Pruned protocol.

4.1 Experimental Setup

The key parameters for our experiments are summarized in Table 4 and detailed below.

Architectures and Fault Model. We evaluated four ResNet architectures (ResNet-20, -32, -44, -56) trained on CIFAR-10 using 4-, 6-, and 8-bit symmetric integer quantization. Table 5 makes their capacity, storage footprint, and train–test behavior explicit. The models range from 272,474 to 855,770 parameters; even the largest configuration has a theoretical packed footprint below 0.83 MiB. Across all architecture–precision combinations, the train–test gap ranges from 5.328 to 7.414 percentage points. At 6 and 8 bits, increasing depth from ResNet-20 to ResNet-56 raises training accuracy toward 99.6% while test accuracy remains close to 92%, and the corresponding gap increases by approximately 1.0–1.6 percentage points. Thus, the deeper models exhibit a somewhat larger generalization gap, but not an abrupt deterioration in test accuracy. These descriptive gaps provide an explicit indicator of possible overfitting; they do not, by themselves, establish its absence. The lower gaps at 4 bits should also be read together with the lower training and test accuracies, since stronger quantization reduces fit as well as generalization performance.

Table 4. Experimental Setup Summary.

Component	**Configuration**
Architectures	ResNet-20, ResNet-32, ResNet-44, ResNet-56
Dataset	CIFAR-10 (quantized-model test accuracy: 86.86%–92.62%; Table 5)
Quantization	4-, 6-, and 8-bit symmetric quantization
Fault model	Class.: MSB (fixed); Regr.: variable bit index
VM type	Extra Trees (classifier) / Random Forest (regressor)
VM objective	Maximize F_β (class.) / predict continuous `acc_drop` (regr.)
BFA algorithms	Greedy Bit Search (GBS), Evolutionary Bit Search (EBS); PBS where reported
Max flip budget	Version/algorithm specific: EBS old 9/6/12/16; EBS new 9/6/12/10; GBS old/new 15/15/22/22 (R-20/32/44/56); regression GBS: at most Top-120
Pre-search filtering	VM-guided Top-120 tuples $c_k = (\ell_k, j_k, b_k)$; common to both variants
Post-search refinement	Standard: none; Pruned: backward elimination after a successful GBS/EBS search
Failure metric	Minimum number of flips (N_{flips}) to reach accuracy $< 11\%$

Dataset scope and external validity. The evaluated ResNet-20/32/44/56 models belong to the compact CIFAR-specific family introduced by He et al. [He *et al*., 2016], rather than to ImageNet-scale ResNet variants transferred unchanged to small images. CIFAR-10 nevertheless limits the evidence to low-resolution, ten-class classification. We did not evaluate ImageNet or another larger-scale dataset in this revision and do not claim that the reported numerical effectiveness generalizes to such settings. The four-architecture, three-precision design preserves coverage of the corresponding CIFAR-10 baseline settings, but is not a cross-dataset generalization experiment. Because the VM learns from dataset- and checkpoint-dependent fault observations, an extension requires additional quantized checkpoints, SFI data, feature extraction, VM fitting, candidate-pool calibration, and attack evaluation with a class-count-appropriate success criterion. Evaluation on larger-scale datasets and architectures remains an important next step.

The fault model implementation differs by approach:

- Classification-Based: Assumes a worst-case scenario, targeting strictly the MSB via the `IntFixedBitFlip` mode of the MRFI multiresolution fault-injection framework.
- Regression-Based: Leverages the granularity of the SFI training paradigms (e.g., BIT-ONE, BIT-ITER). Here, the fault model is flexible, allowing bit-flips at variable positions (not restricted to the MSB) based on the spe-

Table 5. Capacity, theoretical packed size, and baseline accuracy of the quantized CIFAR-10 models. Total parameters include all model parameters. Dividing these counts by the 50,000 images in the standard CIFAR-10 training split gives 5.45, 9.34, 13.23, and 17.12 parameters per training image for ResNet-20/32/44/56, respectively; these ratios are descriptive, not standalone tests for overfitting. Packed size accounts for quantized Conv/Linear weights at the indicated precision and stores the remaining non-quantized parameters in 32-bit floating-point (FP32) format; it is therefore an implementation-independent storage estimate rather than the size of a framework checkpoint. The generalization gap is training accuracy minus test accuracy in percentage points (pp).

Architecture	Precision	Parameters	Packed size (MiB)	Train acc. (%)	Test acc. (%)	Gap (pp)
ResNet-20	4-bit	272,474	0.1352	92.188	86.860	5.328
	6-bit	272,474	0.1998	97.062	91.270	5.792
	8-bit	272,474	0.2644	97.290	91.030	6.260
ResNet-32	4-bit	466,906	0.2309	94.618	88.520	6.098
	6-bit	466,906	0.3416	99.018	92.070	6.948
	8-bit	466,906	0.4524	99.182	92.190	6.992
ResNet-44	4-bit	661,338	0.3266	96.770	89.780	6.990
	6-bit	661,338	0.4835	99.464	92.430	7.034
	8-bit	661,338	0.6403	99.560	92.620	6.940
ResNet-56	4-bit	855,770	0.4223	97.164	90.240	6.924
	6-bit	855,770	0.6253	99.424	92.010	7.414
	8-bit	855,770	0.8283	99.588	92.320	7.268

cific bit index associated with the predicted degradation magnitude.

Vulnerability Model. We evaluated two distinct modeling strategies. First, a classification VM uses Extra Trees to model the binary outcome ("Vulnerable" versus "Not Vulnerable") and is assessed through the F_β score ($\beta \in \{0.5, 1.0, 2.3, 3.0\}$). Second, a regression VM uses a scikit-learn Random Forest to predict continuous `acc_drop`. The regressor has 400 trees, maximum depth 10, three candidate features per split, minimum split size 10, and random seed 42. Its evaluation uses a single random 70/30 hold-out split and reports mean absolute error (MAE), root-mean-square error (RMSE), and R^2; the final pipeline is refitted on all available campaign data after hold-out evaluation. For ranking, the classifier uses the predicted class probability, whereas the regressor uses the predicted accuracy drop.

BFA Search Algorithms and Variants. All evaluated classification attack implementations sort VM outputs and retain Top-120 candidates. This value was selected after a preliminary exploration of candidate-pool sizes $N \in \{50, 70, 120, 200\}$. Across the four tested ResNet architectures, the preliminary results indicated that $N = 120$ offered a practical balance between coverage of potentially vulnerable candidates and search cost. We therefore used the same N for all architectures to maintain a comparable protocol; architecture-specific optimization of N was outside this experiment. EBS represents each retained (ℓ, j) row by one binary gene; because the bit is fixed globally to the MSB for a given quantization, each active gene maps one-to-one to a concrete flip tuple $c_k = (\ell_k, j_k, q-1)$. GBS evaluates every remaining candidate as the next addition on its internal subset and chooses the strict minimum-accuracy trial. PBS is included only in comparisons that explicitly report it.

EBS uses a population of 40, 70 generations, four elites, tournament size three, initial activation probability 0.05, and nominal mutation rate 0.025. Its fitness is the subset baseline-accuracy drop, $A_{\text{sub}}(\emptyset) - A_{\text{sub}}(\mathcal{S})$, with no cardinality penalty. The old and new EBS codes differ in subset size, mutation implementation, deterministic controls, pruning, and the ResNet-56 budget. GBS uses different architecture-specific caps from EBS (Table 3). The evaluated GBS searches traverse the cap even after full-test accuracy falls below 11%; the Pruned implementation then applies backward elimination if the capped solution is successful.

MLBFA-Standard denotes results obtained without backward elimination. MLBFA-Pruned denotes results from the later campaign in which successful GBS/EBS outputs could be eliminated. This campaign label does not imply that all other settings were held fixed. Table 6 and Table 7 report the confirmed protocol differences. When using Regression ranking, the search algorithms respect the candidate-specific position b_k, whereas Classification fixes b_k to the MSB.

Evaluation Metrics. The primary measure of attack compactness is the number of bit-flips (N_{flips}) in a successful solution that reduces Top-1 accuracy below the 11% near-chance threshold defined in Section 3; smaller values denote more compact successful attacks. We analyze the influence of the prediction paradigm (Classification vs. Regression) and quantization precision on this outcome.

Computational-Cost Limitation. A comparative runtime evaluation of Greedy, Evolutionary, and Progressive Bit Search was not performed; therefore, the reported cardinality gains do not establish computational speedup. Backward elimination may require additional network evaluations, so a smaller final flip set does not necessarily imply a shorter execution time.

4.2 VM Evaluation

How predictors affect the VM and the attack The feature definitions and their extraction granularities are specified in Tables 1 and 2. After preprocessing, the tree ensemble maps each candidate's feature vector to a vulnerability score. These scores determine the composition and ordering of the Top-N candidate pool, from which the search algorithm evaluates concrete flips and their combinations. Predictors therefore affect the attack indirectly: a change in score may move a candidate into, out of, or within the pool, potentially changing the solution found and its final cardinality. Because both the tree ensemble and the subsequent search are nonlinear, a large feature value or frequent use in tree splits does not by itself establish a causal improvement in attack effectiveness.

Table 6. Comparison of the EBS experimental protocols used in the classification campaigns.

Setting	MLBFA-Standard	MLBFA-Pruned
Shared settings		
Core hyperparameters	Top-120; $\beta \in \{0.5, 1.0, 2.3, 3.0\}$; population 40; 70 generations; 4 elites; tournament size 3; initial activation probability 0.05; mutation rate 0.025; one-point crossover.	
Objective and evaluation	Subset baseline-accuracy drop, without a cardinality penalty; full-test evaluation of the generation best only after strict subset-fitness improvement; stop at $< 11\%$ accuracy or after 70 generations.	
Configuration-specific settings		
Fitness subset and reproducibility	1,000 images sampled without replacement for each configuration; seed 0	256 images sampled once for each quantization setting and shared across its configurations; seed 0 with deterministic controls for PyTorch and NVIDIA CUDA
Mutation operator	Sequential mutation using Python's `random` module	Vectorized NumPy mutation followed by flip-cap repair
Maximum flips (ResNet-20/32/44/56)	9 / 6 / 12 / 16	9 / 6 / 12 / 10
Post-search step	None	Backward elimination after a successful final solution

Table 7. Comparison of the GBS experimental protocols used in the classification campaigns.

Setting	MLBFA-Standard	MLBFA-Pruned
Shared settings		
Domain, seed, and budget	Descending VM score; Top-120; $\beta \in \{0.5, 1.0, 2.3, 3.0\}$; nominal seed 0; 1,000-image subset; maximum flips 15 / 15 / 22 / 22 for ResNet-20/32/44/56, respectively.	
Selection and evaluation	Evaluate every remaining one-candidate extension; select the first strict subset-accuracy minimum; evaluate every selected prefix on the full test set; success at $< 11\%$ accuracy.	
Configuration-specific setting		
Post-search step	None	Backward elimination on the full test set after a successful capped solution

As a scoped empirical check, a separate Random-Forest explainable artificial intelligence analysis was conducted on the 8-bit ResNet-20 BIT-ONE dataset. For 100 held-out instances, abductive explanations and Local Interpretable Model-Agnostic Explanations (LIME) achieved a mean Jaccard overlap of 0.675, with depth and log-SNIP appearing in 100 and 96 instances, respectively, under both methods. These results identify recurring local decision evidence for this diagnostic model. Estimating the causal contribution of individual predictors to final attack cardinality would additionally require controlled feature-group ablation of the complete Extra-Trees attack pipeline.

Impact of Recall and Precision on Attack Outcomes The effectiveness of the entire MLBFA framework depends on the quality of the ranking generated by the VM. The central question, therefore, is what is the ideal trade-off between Precision and Recall for the VM to produce the most effective ranking? The initial hypothesis, based on protection-oriented works like [Traiola *et al*., 2023a], suggests that Recall should be maximized ($\beta \gg 1$), as the cost of failing to identify a vulnerable parameter (a false negative) is extremely high.

However, our investigation starts from a second hypothesis: for an *attack* that uses search algorithms (EBS, PBS, GBS), an excess of false positives (low precision) can pollute the candidate pool, divert the search to local optima, and consequently result in a suboptimal attack. To test this hypothesis, the F-beta Score (F_β) was used to systematically explore this trade-off, with β values ranging from 0.5 (favoring precision) to 3.0 (favoring recall).

Table 8 presents the complete results of this analysis, showing the final network accuracy for each search algorithm under VMs trained with different β values. The data conclusively confirm our second hypothesis: there is no universally optimal β value. The ideal VM configuration is highly dependent on the scenario, i.e., the specific combination of network architecture, its bit precision, and, notably, the search algorithm employed.

This dependency is clearly visible in the results:

- *Pro-Recall Scenarios:* For ResNet-32 with 4 bits, the table shows that the best overall result (accuracy of 9.76%) was achieved with a VM trained with $\beta = 3.0$, which heavily favors Recall. In this case, the fixed-size Top-120 pool did not become broader; rather, its composition provided greater coverage of candidates predicted to be vulnerable, even at the cost of more false positives, and allowed EBS to find a more lethal combination of flips.
- *Pro-Precision Scenarios:* In contrast, for ResNet-32 with 6 bits, the most devastating attack (accuracy of 10.24%) was obtained jointly with $\beta = 3.0$ on both PBS and GBS columns, while EBS achieved its lowest at $\beta = 2.3$ (10.26%). For ResNet-20 with 8 bits, a fixed-size pool composition with fewer predicted false positives also favored low-β configurations, with PBS/GBS reaching 10.45% at $\beta = 0.5$ and EBS achieving 10.36% at $\beta = 1.0$.
- *Impact on Search Algorithm:* The choice of β influences which search algorithm becomes more effective. For ResNet-20 with 4 bits, for example, the best overall result came from the combination of $\beta = 3.0$ with

Table 8. Impact of the F-score β parameter on the final post-attack accuracy (%). Each value represents the final network accuracy. Boldface indicates the lowest final accuracy in each scenario (row), whether or not that attempt crossed the 11% success threshold.

		EBS				**PBS**				**GBS**			
Architecture	*Quant.*	$\beta = 0.5$	$\beta = 1.0$	$\beta = 2.3$	$\beta = 3.0$	$\beta = 0.5$	$\beta = 1.0$	$\beta = 2.3$	$\beta = 3.0$	$\beta = 0.5$	$\beta = 1.0$	$\beta = 2.3$	$\beta = 3.0$
ResNet-20	4 bits	11.90	13.84	10.99	10.95	19.10	17.80	17.80	12.91	15.05	12.36	12.68	**10.53**
	6 bits	10.76	11.51	10.73	10.87	11.44	11.78	10.96	11.16	10.42	10.70	10.98	**10.29**
	8 bits	10.92	**10.36**	12.40	10.97	10.45	10.94	10.91	11.04	10.45	10.94	10.91	10.58
ResNet-32	4 bits	22.31	10.95	13.49	**9.76**	10.44	10.44	10.44	10.44	10.44	10.44	10.44	10.44
	6 bits	10.92	10.42	10.26	12.75	12.49	13.73	12.49	**10.24**	10.67	10.62	10.67	**10.24**
	8 bits	10.86	13.75	14.01	14.39	13.17	13.17	13.17	15.05	10.73	**10.34**	10.57	10.95
ResNet-44	4 bits	12.40	14.42	11.78	13.48	15.69	15.69	21.48	22.26	**10.58**	10.59	10.87	10.64
	6 bits	25.23	16.56	17.47	21.75	32.23	32.23	32.23	32.23	12.29	**12.19**	12.87	12.23
	8 bits	12.15	12.92	12.39	13.44	39.62	39.62	39.62	39.62	11.16	**10.94**	11.25	10.99
ResNet-56	4 bits	**10.39**	10.83	10.84	11.16	10.94	10.87	10.90	10.85	10.94	10.87	10.90	10.85
	6 bits	10.98	9.78	10.90	10.98	**8.86**	**8.86**	**8.86**	**8.86**	**8.86**	**8.86**	**8.86**	**8.86**
	8 bits	9.06	9.23	9.51	10.78	**8.99**	**8.99**	10.83	10.83	**8.99**	**8.99**	10.83	10.83

Table 9. Selected attack outcomes and solution cardinalities: Smallest observed number of bit-flips (N_{flips}) among successful attacks (accuracy $< 11\%$) in each scenario. Beta, Recall, and Precision values refer to the VM that guided the respective attack. †For ResNet-44 6-bit, no attack within the maximum flip budget reached the $<$11% threshold; the attempt with the lowest residual accuracy is reported.

Architecture	*Quantization*	*Search Algorithm*	N_{flips}	*Final Accuracy (%)*	*Beta (F-score)*	*Recall (VM)*	*Precision (VM)*
ResNet-20	4	EBS	9	10.99	2.3	0.7600	0.1583
	6	EBS	9	10.76	0.5	0.2857	0.8000
	8	EBS	9	10.77	0.5	0.2778	0.8333
ResNet-32	4	EBS	6	9.76	3.0	0.9500	0.2774
	6	EBS	6	10.26	2.3	0.9121	0.5533
	8	EBS	6	10.24	0.5	0.7826	0.9000
ResNet-44	4	GBS	11	10.58	0.5	0.6933	0.7429
	6	GBS	22†	12.19	1.0	0.7949	0.7045
	8	GBS	17	10.90	0.5	0.8529	0.6304
ResNet-56	4	GBS	10	10.87	1.0	0.8333	0.5769
	6	GBS	11	8.86	0.5	0.4706	1.0000
	8	GBS	10	10.77	2.3	0.7727	0.5000

the GBS algorithm (10.53% accuracy), even though the EBS algorithm performed better with $\beta = 2.3$ (10.99% accuracy).

This detailed analysis, supported by Table 8, validates the thesis that the hyperparameter optimization step of the VM is a crucial part of our framework. A one-size-fits-all approach to the Precision-Recall trade-off would lead to significantly inferior attack performance in most scenarios.

Final Accuracy across Quantization Levels Figure 2 summarizes the reported final post-attack accuracy using the balanced $K = 2{,}024$ internal test-subset stratum of the reconstructed dataset. The complete dataset contains 240 unique attacks across $K \in \{1{,}024, 2{,}024, 10{,}000\}$, but the 1,024- and 10,000-image conditions are available only for 8-bit quantization. Pooling all three conditions would therefore give 8-bit experiments three times as many observations and would confound quantization with internal subset size. Restricting this figure to $K = 2{,}024$ provides the complete factorial comparison: three quantization levels, four architectures (ResNet-20/32/44/56), three search algorithms, and four β values. After removing the two-class representational duplication in the merged dataset, each $(q, \text{architecture}, \text{algorithm}, \beta)$ configuration is plotted exactly once, yielding 144 unique attacks and 48 observations per quantization level.

The figure includes both successful and unsuccessful attempts and uses the same reported final-accuracy field for plotting and outcome classification. Under the protocol defined in Section 3, an attempt is successful only when its final full-test accuracy satisfies $A_{\text{test}}(\mathcal{S}) < 11\%$; the dashed line marks this threshold. Points above 11% are unsuccessful attempts retained to characterize the complete outcome distribution. Of the 48 configurations at each precision, 28 reached the target at each of 4, 6, and 8 bits. The corresponding mean final accuracies are 12.36%, 12.80%, and 12.78%, while the medians are 10.84%, 10.91%, and 10.89%. These statistics summarize all attempts rather than only successful attacks.

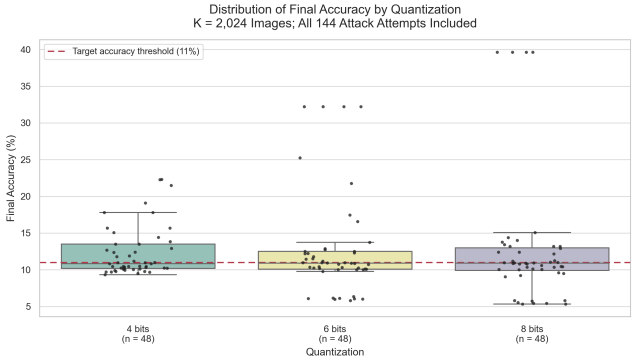

Figure 2. Distribution of reported final full-test accuracy, $A_{\text{test}}(\mathcal{S})$, by quantization for the balanced $K = 2{,}024$ condition (144 unique attacks; 48 per quantization level). Each point represents one architecture–algorithm–β configuration across ResNet-20/32/44/56. Successful and unsuccessful attempts are both included; therefore, values above the dashed 11% threshold are expected and denote attempts that did not meet the success criterion. The $K = 1{,}024$ and $K = 10{,}000$ records are excluded here because they are available only for 8-bit quantization and are analyzed separately in Figure 3.

Impact of Internal Test-Subset Size on Final Attack Accuracy To examine whether reducing the number of images per internal evaluation changes the observed final attack accuracy, we reanalyzed the 8-bit experimental records using $K \in \{1{,}024, 2{,}024, 10{,}000\}$. All three conditions use images from the CIFAR-10 test set; $K = 10{,}000$ corresponds to the complete test set. The merged dataset contains 288 class-level rows (96 per condition), because each attack configuration is represented once for each of the two VM classes. After removing this representational duplication, the analysis contains 144 unique attack configurations (48 per condition) and includes both successful and unsuccessful attempts. Each unique $(K, \text{architecture}, \text{algorithm}, \beta)$ configuration contributes exactly one final-accuracy value to Table 10 and Figure 3. Consequently, observations and aggregate means above 11% represent unsuccessful attempts and do not contradict the success criterion $A_{\text{test}}(\mathcal{S}) < 11\%$.

The means range from 12.78% to 13.50%, the medians from 10.51% to 10.89%, and the distributions overlap substantially (Figure 3). These descriptive results suggest limited sensitivity of central tendency to the tested subset size; formal equivalence would require a dedicated inferential design. Using 1,024 rather than 10,000 images reduces the number of inferences per internal subset evaluation by 89.76%. This percentage quantifies computational count, while end-to-end runtime was not measured.

Table 10. Final full-test accuracy statistics (%) by internal test-subset size for the 8-bit experiments after removing the two-class representational duplication. Each condition contains 48 unique attack configurations and includes both successful and unsuccessful attempts. Std. dev. denotes population standard deviation; IQR denotes interquartile range.

Subset Size	*Mean*	*Median*	*Std. dev.*	*IQR*
1024	13.22	10.68	8.56	1.49
2024	12.78	10.89	8.53	3.08
10000	13.50	10.51	8.43	3.16

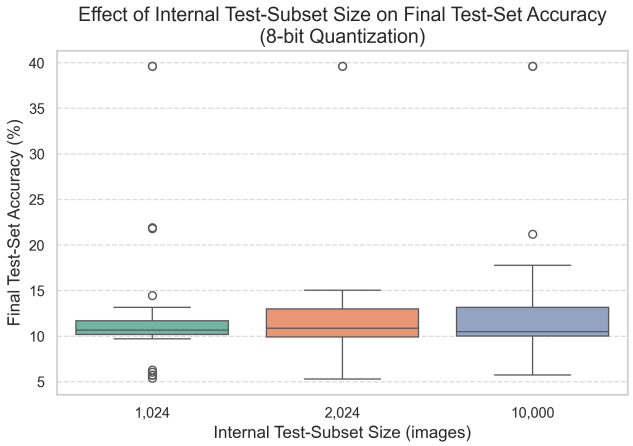

Figure 3. Distribution of final full-test accuracy, $A_{\text{test}}(\mathcal{S})$, by internal test-subset size in the 8-bit experiments after removing the two-class representational duplication (48 unique attack configurations per condition). Successful and unsuccessful attempts are both included; therefore, values above 11% are expected and denote attempts that did not meet the success criterion.

4.3 Classification-Based MLBFA-Standard without Backward Elimination

This subsection reports the classification-based MLBFA-Standard results first published in the 2025 Symposium on Computing Systems Engineering (SBESC) paper [Ronaldo *et al.*, 2025], with the corrected tabulation adopted by the present article. These runs use VM-guided Top-N filtering and BFA search without post-search backward elimination; the reported number of bit-flips (N_{flips}) is therefore the cardinality returned by the search itself. Table 9 summarizes the smallest observed successful attacks for each architecture and quantization combination, identifying the corresponding search algorithm and VM characteristics and explicitly marking the unsuccessful ResNet-44 6-bit case.

The selected Standard results show that VM-guided search can reach the target with small flip sets. In one descriptive example, the reported ResNet-32 selections use 6 flips at each tested precision. These selected previously reported outcomes are retained to document the conference pipeline and the evolution of the framework. They are not used in the primary statistical comparison with Rakin et al. [Rakin *et al.*, 2019], because the available Standard selections do not constitute an observation set aligned with the median–median protocol adopted for that comparison.

Within the selected Standard entries in Table 9, EBS supplies the reported ResNet-20/32 rows and GBS supplies the ResNet-44/56 rows. This descriptive pattern combines variation in architecture depth, algorithm, VM ranking, and random trajectory; isolating their individual contributions would require a controlled factorial design. Notably, the reported Standard configurations struggled on *ResNet-44* at higher quantization (17 flips at 8-bit and no successful attack within the reported 22-flip budget at 6-bit). Section 4.4 separately analyzes configurations that include post-search backward elimination, an operator applied to successful input solutions.

4.4 Classification-Based MLBFA-Pruned with Backward Elimination

This subsection compares selected best outcomes from the independently executed classification-based MLBFA-Standard and MLBFA-Pruned campaigns. In the Pruned variant, successful GBS/EBS outputs are supplied to post-search backward elimination, and the reported cardinality is the final size after that elimination attempt. Table 11 presents the campaign-level comparison. Because the two sides may differ in the supplying algorithm, β, seed policy, subset protocol, and search cap, as documented in Tables 6 and 7, the table compares final outcomes rather than estimating the causal effect of backward elimination.

To verify the operator's set-inclusion property at the implementation level, we examined all campaign-best Pruned cases for which explicit paired before/after GBS data are available. As summarized in Table 12, every recorded case satisfies $|\mathcal{S}_{\text{after}}| \leq |\mathcal{S}_{\text{before}}|$, while the final solution preserves the target-accuracy condition. These selected records provide a consistency check, not an estimate of the operator's average causal effect. EBS is not included in this check because its available records contain final cardinality, final accuracy, and

Table 11. Best independently obtained results from the Standard and Pruned campaigns. Each row compares campaign-level outcomes; the algorithm and β columns report the corresponding protocol differences. †No Standard attack reached $< 11\%$ within the 22-flip GBS cap.

		Standard campaign			**Pruned campaign**		
Architecture	*Bits*	*Search*	β	*Flips*	*Search*	β	*Final flips*
ResNet-20	8	EBS	0.5	9	EBS	1.0	7
	6	EBS	0.5	9	EBS	0.5	8
	4	EBS	2.3	9	GBS	1.0	11
ResNet-32	8	EBS	0.5	6	EBS	1.0	4
	6	EBS	2.3	6	GBS	0.5	4
	4	EBS	3.0	6	GBS	0.5	6
ResNet-44	8	GBS	0.5	17	GBS	1.0	9
	6	GBS	1.0	–†	GBS	2.3	11
	4	GBS	0.5	11	GBS	3.0	8
ResNet-56	8	GBS	2.3	10	EBS	3.0	5
	6	GBS	0.5	11	EBS	2.3	6
	4	GBS	1.0	10	EBS	0.5	7

Table 12. Available paired before/after records of backward elimination. These are all campaign-best Pruned cases for which the experimental data retain explicit pre- and post-elimination results.

Architecture	*Quant.*	β	*Before → after*	*Final acc.*
ResNet-20	4	1.0	$15 \rightarrow 11$	10.91%
ResNet-32	4	0.5	$15 \rightarrow 6$	10.25%
ResNet-32	6	0.5	$15 \rightarrow 4$	10.93%
ResNet-44	4	3.0	$22 \rightarrow 8$	10.11%
ResNet-44	6	2.3	$22 \rightarrow 11$	10.68%
ResNet-44	8	1.0	$22 \rightarrow 9$	10.68%

All displayed records are GBS executions with nominal seed 0. EBS records are excluded because they retain only the final cardinality and an elimination flag, not the pre-elimination cardinality. The table is an implementation-consistency check, not an estimate of the operator's average effect.

an elimination flag, but not the pre-elimination cardinality. For 4-bit ResNet-20, GBS traversed its full 15-flip cap even though the full-test accuracy was already 10.65% after 13 flips; backward elimination was applied only to the capped solution and produced the recorded reduction 15→11.

The apparent 4-bit ResNet-20 inconsistency does not represent a before/after application of backward elimination. The 9-flip Standard result was obtained independently by EBS with $\beta = 2.3$, a 9-flip cap, and final accuracy of 10.99%. The 11-flip Pruned result was obtained by GBS with $\beta = 1.0$, which supplied a 15-flip solution to backward elimination and returned 11 flips with final accuracy of 10.91%. Although both configurations used Top-120 and nominal seed 0, they have different search algorithms and β values, and no shared run or solution identifier links them. The former was the selected Standard campaign best case, whereas the latter was the sole successful result among the eight corresponding Pruned configurations. The latter is not an EBS-Pruned result: every 4-bit ResNet-20 EBS-Pruned configuration is capped at 9 and remains above the 11% target. Thus, the selected Pruned outcome is not smaller than the Standard best case in this configuration, but backward elimination itself reduced its supplied solution from 15 to 11 flips. The corresponding source files and experiment records are identified in the public repository.

The EBS-to-EBS records reinforce this interpretation. At $\beta = 2.3$, Standard EBS reached 10.99% with 9 flips, whereas the later EBS implementation returned 13.81% with 9 flips and `Pruned=False`; backward elimination therefore did not run. Although core settings match (Top-120, cap 9, seed 0, 70 generations, and population 40), the protocols differ in the fitness subset (1,000 versus 256 samples and different random-number consumption) and the mutation operator (sequential Python-random mutation versus vectorized NumPy mutation with cap repair). These differences alter fitness and the evolutionary trajectory, placing this EBS-to-EBS result at the campaign-comparison level; the paired GBS records separately characterize backward elimination.

The 6-bit ResNet-44 entry illustrates the same distinction from another direction. The Standard GBS run reported in the comparison did not succeed within its 22-flip cap (final accuracy 12.19%), whereas a separate Pruned-campaign GBS run with $\beta = 2.3$ supplied a successful 22-flip input and reduced it to 11. Backward elimination did not convert the unsuccessful Standard run into a success.

For descriptive context, the selected Pruned campaign value is lower than or equal to the selected Standard value in 11 of 12 entries. This count summarizes independent campaign-level outcomes, while the paired before/after records in Table 12 characterize the effect of backward elimination. Table 13 presents the campaign minima in a separate exploratory minimum-to-minimum comparison with Rakin et al. [Rakin *et al*., 2019]; the balanced median-based comparison below remains the primary cross-study statistic.

Primary balanced descriptive comparison with Rakin et al. To make the cross-study comparison internally consistent, we use the median number of flips among successful attacks as the primary cardinality statistic on both sides. For MLBFA-Pruned, we restrict the analysis to $K = 2{,}024$, the internal subset size shared by all 4-, 6-, and 8-bit settings. The 8-bit archive additionally contains $K = 1{,}024$ and $K = 10{,}000$ conditions, whereas the 4- and 6-bit archives contain only $K = 2{,}024$; pooling those additional conditions would yield 24 observations for each 8-bit architecture but only eight for each 4- or 6-bit architecture and would confound precision with internal subset size. At $K = 2{,}024$, each architecture–precision pair comprises eight evaluated EBS/GBS–β configurations (two search algorithms and four β values). Because these observations vary both search algorithm and β, they are treated as evaluated configurations rather than repeated iid trials of a fixed algorithm.

Table 14 shows lower conditional MLBFA-Pruned medians in 10 of the 12 architecture–precision settings and higher medians in two. The observed fraction of successful configurations and their dispersion provide context for these me-

Table 13. Exploratory best-observed cross-study context. The Rakin et al. [Rakin *et al*., 2019] column reports the minimum cardinality among the five trials published in their Table 2, whereas the MLBFA-Pruned column reports the smallest successful attack observed among the evaluated EBS/GBS–β configurations. For 8-bit models, the campaign minimum is selected across the $K = 1{,}024$, 2,024, and 10,000 internal-subset strata; only $K = 2{,}024$ is available for 4- and 6-bit models. The selected K is reported explicitly. Because minima depend on the number and composition of evaluated trials/configurations, this table provides exploratory best-case context; the balanced median-based comparison in Table 14 remains primary. N_{flips} is the final solution cardinality. EBS denotes Evolutionary Bit Search and GBS denotes Greedy Bit Search.

Architecture	*Bits*	***K***	*Rakin min.*	*MLBFA-Pruned min.*	*Search*	*Final Accuracy (%)*	*Beta (F-score)*
ResNet-20	4	2024	7	11	GBS	10.91	1.0
	6	2024	8	8	EBS	10.55	0.5
	8	2024	7	7	EBS	10.45	1.0
ResNet-32	4	2024	10	6	GBS	10.25	0.5
	6	2024	9	4	GBS	10.93	0.5
	8	1024	8	4	EBS	10.73	1.0
ResNet-44	4	2024	14	8	GBS	10.11	3.0
	6	2024	13	11	GBS	10.68	2.3
	8	10000	6	9	GBS	10.68	1.0
ResNet-56	4	2024	9	7	EBS	10.43	0.5
	6	2024	16	6	EBS	10.79	2.3
	8	1024	16	5	EBS	10.60	3.0

Table 14. Descriptive cross-study comparison of solution cardinality (N_{flips}) between Rakin et al. [Rakin *et al*., 2019] and MLBFA-Pruned. Rakin statistics are reconstructed from the five trial cardinalities reported in their Table 2; every listed trial reached the paper's below-11% endpoint, so its observed completion fraction is 5/5. To avoid asymmetric pooling of the additional 8-bit subset-size conditions, MLBFA-Pruned is restricted to the balanced $K = 2{,}024$ stratum, comprising eight EBS/GBS–β configurations for each architecture–precision pair. Median, range, and sample standard deviation are computed over successful outcomes; successful trials/configurations and their observed fractions are reported explicitly. N/A denotes a standard deviation that cannot be estimated from one successful configuration. The two studies use distinct search procedures, hyperparameters, candidate construction, execution counts, and budget/stopping specifications.

		Rakin et al. (5 trials)			MLBFA-Pruned (8 configurations)		
Architecture	*Bits*	*Successful*	*Median [range]*	*SD*	*Successful*	*Median [range]*	*SD*
ResNet-20	8	5/5 (100%)	10 [7–17]	3.70	6/8 (75.0%)	8.5 [7–14]	2.64
	6	5/5 (100%)	11 [8–13]	2.30	5/8 (62.5%)	8 [8–13]	2.24
	4	5/5 (100%)	7 [7–12]	2.17	1/8 (12.5%)	11 [11–11]	N/A
ResNet-32	8	5/5 (100%)	12 [8–31]	9.40	7/8 (87.5%)	6 [5–10]	1.77
	6	5/5 (100%)	12 [9–23]	5.59	7/8 (87.5%)	5 [4–7]	1.00
	4	5/5 (100%)	14 [10–17]	2.61	5/8 (62.5%)	6 [6–6]	0.00
ResNet-44	8	5/5 (100%)	11 [6–22]	5.94	7/8 (87.5%)	12 [11–20]	3.18
	6	5/5 (100%)	15 [13–17]	1.79	3/8 (37.5%)	13 [11–16]	2.52
	4	5/5 (100%)	15 [14–50]	15.79	7/8 (87.5%)	11 [8–14]	2.34
ResNet-56	8	5/5 (100%)	18 [16–22]	2.83	6/8 (75.0%)	9.5 [6–12]	1.97
	6	5/5 (100%)	17 [16–21]	2.35	6/8 (75.0%)	8.5 [6–10]	1.37
	4	5/5 (100%)	21 [9–24]	5.94	4/8 (50.0%)	11.5 [7–17]	4.11

dians. For example, the ResNet-32 8-bit median of 6 flips is supported by seven successful configurations out of eight, whereas the ResNet-20 4-bit median of 11 flips is based on one successful configuration out of eight. Rakin et al.'s Table 2 supplies five below-11% trial outcomes for every setting, hence the displayed 5/5 observed completion fraction. The comparison is descriptive because the studies differ in search procedure, hyperparameterization, execution count, candidate construction, and budget/stopping specification.

4.5 Behavioral Characterization of Classification-Guided Attacks

Layer and bit locations. To identify where the successful classification-guided attacks acted, we joined each successful experiment in the consolidated Standard and 2026 campaign datasets to its recorded flip list. The duplicated class-metric rows in the Standard dataset were first collapsed by the experimental key (internal subset size, quantization, architecture, search algorithm, and β). For sequential GBS/PBS records, we retained the shortest logged prefix whose full-test accuracy satisfied $A_{\text{test}}(\mathcal{S}) < 11\%$; where only a final EBS or post-elimination solution was recorded, that complete success-

ful solution was used. This procedure yields 253 successful attacks—141 Standard records and 112 records from the 2026 campaign—and 2,537 flips. Table 15 reports their layer locations. These pooled descriptive counts give more weight to attacks containing more flips; parameter-normalized vulnerability rates would additionally account for differences in layer size.

Table 15. Layer locations of the 2,537 flips in the first reconstructable successful sets from 253 recorded classification-guided attacks. "Attacks hitting layer" counts attacks containing at least one flip in that layer; because one attack may reach multiple layers, this column is not additive. Flip shares are pooled descriptive percentages and are not normalized by the number of parameters in each layer.

Layer	*Layer type*	*Flips*	*Flip share (%)*	*Attacks, n (%)*
`layer2.0.shortcut.0`	Shortcut convolution	1,282	50.53	223 (88.14)
`conv1`	Stem convolution	936	36.89	149 (58.89)
`layer1.2.conv1`	Residual convolution	256	10.09	70 (27.67)
`layer1.0.conv1`	Residual convolution	44	1.73	22 (8.70)
`layer3.0.shortcut.0`	Shortcut convolution	18	0.71	9 (3.56)
`linear`	Linear classifier	1	0.04	1 (0.40)

Aggregating the same records by layer type, shortcut convolutions account for 1,300 flips (51.24%) and are reached by 224 of the 253 successful attacks (88.54%). The stem convolution accounts for 936 flips (36.89%) and appears in 149 attacks (58.89%); residual convolutions account for 300 flips (11.82%) and appear in 92 attacks (36.36%); and the linear classifier accounts for one flip (0.04%) in one attack (0.40%). The two most frequent concrete layers, `layer2.0.shortcut.0` and `conv1`, therefore contain 87.42% of all reconstructed successful-set flips. This concentration identifies where the evaluated searches acted, but it does not by itself establish an intrinsic causal susceptibility of layer types because the candidate ranking, candidate-pool composition, layer sizes, and search trajectories were not controlled for this comparison.

The bit location is fixed by the classification-based fault protocol rather than selected by these searches. With zero-based indexing from the least significant bit, every candidate uses $b = q - 1$, i.e., the most significant/sign bit: $b = 3$, 5, and 7 for 4-, 6-, and 8-bit quantization, respectively. Table 16 makes this distribution explicit. Toggling these bits changes the signed integer code by $|\Delta q| = 2^b$ (8, 32, and 128, respectively), while the dequantized change is $|\Delta w| = s_\ell 2^b$ for the executed layer scale s_ℓ. The available records omit the model checkpoints and per-layer scales required to recover weight-space magnitudes. The counts across $b = 3, 5, 7$ therefore summarize successful-set flips under the protocol-fixed MSB/sign-bit rule; evaluating relative bit-plane vulnerability would require admissible alternatives within each quantization level.

Table 16. Bit positions in the 2,537 reconstructed successful-set flips. Bit indices are zero-based from the LSB. All are MSB/sign-bit flips by construction of the classification-based protocol.

Quant. (bits)	*Bit index b*	$\lvert\Delta q\rvert$	*Flips*	*Share (%)*
4	3	8	440	17.34
6	5	32	453	17.86
8	7	128	1,644	64.80

Progressive accuracy degradation. The final cardinality alone does not show how rapidly accuracy deteriorates as faults accumulate. We therefore reconstruct a fixed representative slice for which every ordered prefix was recorded: MLBFA-Standard GBS with 8-bit quantization, $K = 10{,}000$, and $\beta = 0.5$, for all four architectures. This slice holds the search algorithm, precision, internal-set size, and VM hyperparameter fixed; it was not selected separately for the best outcome of each architecture. Let $\mathcal{S}_0 = \emptyset$ and $\mathcal{S}_k = \{c_1, \ldots, c_k\}$ denote the first k flips selected by GBS. Figure 4 reports the recorded full-test accuracy $A_{\text{test}}(\mathcal{S}_k)$ for every $k \in \{0, \ldots, 10\}$.

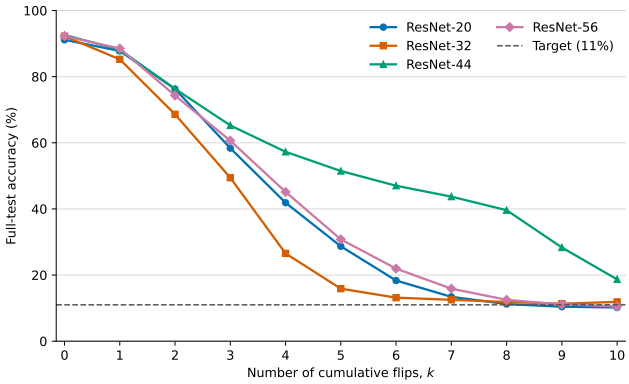

Figure 4. Progressive full-test accuracy for the first ten cumulative GBS flips in a fixed representative MLBFA-Standard slice (8-bit quantization, $K = 10{,}000$, and $\beta = 0.5$). The dashed line is the 11% success threshold; $k = 0$ is the quantized-model baseline. Each curve represents one recorded ordered trajectory.

All four baselines lie between 91.14% and 92.62%, but the degradation rate is architecture dependent. At $k = 10$, accuracy is 10.19% for ResNet-20, 11.88% for ResNet-32, 18.70% for ResNet-44, and 10.52% for ResNet-56. Thus, ResNet-20 first crosses the 11% threshold at $k = 9$ and ResNet-56 at $k = 10$, whereas ResNet-32 and ResNet-44 remain above it within the displayed range. The prefixes are cumulative, but their full-test accuracies need not be strictly monotonic: for ResNet-32, accuracy changes from 11.34% at $k = 9$ to 11.88% at $k = 10$. This small reversal is compatible with interactions among simultaneous faults and with GBS choosing the next flip by accuracy on its internal subset rather than by the plotted full-test value. The figure characterizes progressive behavior within this fixed slice; broader degradation patterns across architectures require additional controlled configurations.

4.6 Regression-Based Vulnerability Ranking

Complementing the classification-based analysis, we investigated an alternative regression-based strategy, where the VM is trained to predict the exact magnitude of accuracy degradation caused by a bit-flip. Table 17 details the performance of EBS, implemented with a genetic algorithm, and GBS under this guidance. The "SFI Paradigm" column indicates the source of the training data—*Bit-Wise* (BIT-ONE, BIT-ITER) or coarser structural approaches (LAYER, NET)—allowing us to assess how data granularity influences the quality of the regression.

Table 17. Performance comparison of regression-based vulnerability ranking using different SFI paradigms (BIT-ITER, BIT-ONE, LAYER, NET) for training data generation. EBS uses the architecture-specific limits in Table 3; GBS traverses at most Top-120. For successful attacks, *Bits* is the final cardinality after backward elimination; 120 with *Rch=No* denotes GBS candidate-pool exhaustion. In the headings, *Err* denotes target error, *Acc.* denotes accuracy, and *Rch* denotes whether the target was reached.

		EBS			GBS					EBS			GBS		
Err	SFI	Bits	Acc.	Rch	Bits	Acc.	Rch	Err	SFI	Bits	Acc.	Rch	Bits	Acc.	Rch
ResNet-20								**ResNet-44**							
0.5%	BIT-ITER	8	0.1094	Yes	8	0.1025	Yes	0.5%	BIT-ITER	12	0.6074	No	120	0.5869	No
0.5%	BIT-ONE	9	0.1895	No	8	0.1025	Yes	0.5%	BIT-ONE	11	0.2803	No	120	0.7852	No
0.5%	LAYER	9	0.1221	No	12	0.1035	Yes	0.5%	LAYER	12	0.1934	No	36	0.1094	Yes
0.5%	NET	7	0.1094	Yes	9	0.1064	Yes	0.5%	NET	3	0.8125	No	120	0.8994	No
1%	BIT-ITER	6	0.8652	No	120	0.8125	No	1%	BIT-ITER	1	0.9023	No	120	0.9004	No
1%	BIT-ONE	9	0.1162	No	9	0.1064	Yes	1%	BIT-ONE	12	0.2734	No	47	0.1094	Yes
1%	LAYER	9	0.1514	No	10	0.1064	Yes	1%	LAYER	12	0.3027	No	120	0.1611	No
1%	NET	9	0.1982	No	24	0.1094	Yes	1%	NET	12	0.5117	No	120	0.7295	No
5%	BIT-ITER	1	0.8994	No	120	0.8838	No	5%	BIT-ITER	2	0.9004	No	120	0.8135	No
5%	BIT-ONE	9	0.2236	No	120	0.1738	No	5%	BIT-ONE	12	0.3037	No	120	0.3174	No
5%	LAYER	9	0.2715	No	45	0.1084	Yes	5%	LAYER	12	0.2832	No	120	0.1670	No
5%	NET	1	0.9053	No	120	0.9053	No	5%	NET	1	0.9023	No	120	0.9023	No
ResNet-32								**ResNet-56**							
0.5%	BIT-ITER	6	0.2266	No	15	0.1074	Yes	0.5%	BIT-ITER	10	0.3467	No	120	0.2178	No
0.5%	BIT-ONE	6	0.3838	No	33	0.1094	Yes	0.5%	BIT-ONE	8	0.3125	No	120	0.4053	No
0.5%	LAYER	6	0.3604	No	29	0.1094	Yes	0.5%	LAYER	10	0.1602	No	32	0.1094	Yes
0.5%	NET	6	0.3799	No	120	0.6348	No	0.5%	NET	1	0.9131	No	120	0.9131	No
1%	BIT-ITER	6	0.5791	No	120	0.5752	No	1%	BIT-ITER	9	0.1064	Yes	10	0.0928	Yes
1%	BIT-ONE	6	0.1143	No	11	0.1094	Yes	1%	BIT-ONE	10	0.2549	No	120	0.1865	No
1%	LAYER	6	0.1523	No	23	0.1094	Yes	1%	LAYER	10	0.2080	No	29	0.1094	Yes
1%	NET	1	0.8994	No	120	0.9053	No	1%	NET	5	0.8828	No	120	0.8457	No
5%	BIT-ITER	1	0.8975	No	120	0.9004	No	5%	BIT-ITER	1	0.9131	No	120	0.9131	No
5%	BIT-ONE	6	0.1279	No	10	0.1035	Yes	5%	BIT-ONE	9	0.2734	No	120	0.2275	No
5%	LAYER	6	0.3535	No	120	0.1494	No	5%	LAYER	10	0.2520	No	27	0.1094	Yes
5%	NET	1	0.8994	No	120	0.8994	No	5%	NET	10	0.3262	No	120	0.4307	No

Table 18. Best attack scenarios for Regression-based ranking per architecture. Configuration format: Algorithm / SFI Paradigm (Target Error). *Bits* is the final cardinality after the MLBFA-Pruned backward-elimination attempt.

Architecture	Configuration	Bits	Final Acc.
ResNet-20	EBS / NET (0.5%)	7	0.1094
ResNet-32	GBS / BIT-ONE (5%)	10	0.1035
ResNet-44	GBS / LAYER (0.5%)	36	0.1094
ResNet-56	EBS / BIT-ITER (1%)	9	0.1064

The regression-guided EBS and GBS attacks were executed under MLBFA-Pruned. EBS uses the EBS-new budgets in Table 3; its fitness is computed on a 256-sample internal subset, while successful solutions and pruning are evaluated on the stored full 1,024-sample test set. Regression-guided GBS has no EBS-style architecture cap: it selects from Top-120 and stops when the target is reached or that pool is exhausted. Thus, a GBS entry of 120 with *Rch=No* denotes exhaustion without a successful attack, whereas values such as 36 are valid GBS solution cardinalities. For either search, a successful solution with more than one flip is supplied to the same iterative backward-elimination operator used in the classification-based attacks.

Accordingly, for successful regression-guided EBS and GBS attacks, *Bits* reports the final cardinality after backward elimination; unsuccessful rows do not trigger elimination.

A descriptive comparison with the classification results in Section 4.4 shows larger selected solution cardinalities for regression-based ranking in these experiments. As summarized in Table 18, the selected ResNet-32 result uses 10 flips under regression ranking versus 4 under classification; the corresponding values are 36 versus 8 for ResNet-44 and 9 versus 5 for ResNet-56. These are best-case comparisons across configurations, not paired estimates or measurements of computational cost.

Table 17 also records numerous configurations that did not reach the target, particularly for deeper architectures, and successful GBS configurations can require many flips. Within the reported data, classification ranking supplied smaller selected best cases than regression ranking for three of the four architectures. This descriptive pattern combines differences in learning objective, training-data paradigm, search algorithm, and protocol; isolating their individual contributions would require a controlled design.

5 Conclusion

This work extends the MLBFA framework introduced at SBESC 2025 [Ronaldo *et al*., 2025]. The inherited components comprise the classification VM, 21-feature candidate representation and its predefined screening, VM-guided Top-N filtering, GBS/EBS/PBS searches, four ResNet architectures, three quantization levels, and the F_β and validation-size analyses. The extension contributes backward elimination in MLBFA-Pruned, regression-based bit-specific ranking with additional SFI paradigms, and expanded paired, outcome, and protocol analyses. VM-guided Top-N filtering precedes search, while backward elimination is a distinct post-search operator. The Standard and Pruned campaign-level best cases use different search algorithms, β values, subset handling, budgets, and code paths. Their final outcomes—including 10 versus 5 flips for independently selected 8-bit ResNet-56

configurations—therefore provide a descriptive campaign-level comparison. The primary descriptive comparison with Rakin et al. uses median successful-attack cardinality on both sides and the balanced MLBFA-Pruned $K = 2{,}024$ stratum, together with observed campaign success fractions and dispersion; MLBFA-Pruned yielded a lower median in 10 of 12 architecture–precision settings. The complementary best-observed comparison yielded lower minima in eight settings, equal minima in two, and higher minima in two under the evaluated campaign coverage.

The paired analysis establishes the operator-level result. Backward elimination reduced every recorded GBS input solution; in the 4-bit ResNet-20 pair, it reduced 15 to 11 flips. The separate Standard EBS campaign supplied the reported 9-flip best case with $\beta = 2.3$, while the paired GBS result quantifies the reduction applied to its own 15-flip input. The layer-location analysis further shows that shortcut and stem convolutions contain 51.24% and 36.89%, respectively, of the 2,537 reconstructed flips from 253 successful classification-guided attacks; all such attacks toggle the protocol-fixed MSB/sign bit. In the fixed 8-bit GBS slice used to expose progressive behavior, ten cumulative flips reduced full-test accuracy to 10.19%, 11.88%, 18.70%, and 10.52% for ResNet-20, -32, -44, and -56, respectively, revealing architecture-dependent degradation rates. Classification ranking supplied smaller selected best cases than regression ranking for three of four architectures, and the best VM trade-off remained scenario dependent. The measured effect of backward elimination applies to final solution cardinality; pre-search candidate filtering, explored search space, and runtime are separate stages and metrics. Future evaluations should serialize candidate-file hashes, seeds, pre-elimination solutions, pre/post-elimination records, attempted/successful run counts, exact protocol versions, model checkpoints, and per-layer quantization scales so that operator effects and weight-space perturbation magnitudes can be reported reproducibly.

Future work will evaluate larger-scale datasets and architectures, including ImageNet, and measure the runtime trade-offs of search and backward elimination under documented execution conditions. Further extensions include targeted attacks, multiple-bit upsets, and the use of vulnerability maps to guide selective hardware protection schemes, such as error-correcting codes. The present numerical conclusions remain restricted to the evaluated CIFAR-10 configurations.

Declarations

Authors' Contributions

José Ronaldo: Conceptualization, Methodology, Software, Investigation, Data Curation, Writing–original draft, Visualization.

Otávio Alcântara de Lima: Conceptualization, Methodology, Supervision, Validation, Formal analysis, Writing–review & editing.

Competing interests

The authors declare that they have no competing interests.

Funding

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001.

Availability of data and materials

The source code, experiment records, consolidated configurations, random-seed policy, dependency specifications, file manifest, and scripts used to regenerate the released tables and figures are publicly available at `https://github.com/JRonaldoFerreira/MLBFA`. The repository provides file checksums and identifies the model checkpoints, candidate rankings, and statistical fault-injection data required for end-to-end campaign execution. Because these large external artifacts are not distributed with the code release, their availability and the resulting scope of reproduction are stated in the repository.

References

Agarwal, U. K., Chan, A., and Pattabiraman, K. (2022). LLTFI: Framework-agnostic fault injection for machine learning applications. In *2022 IEEE 33rd International Symposium on Software Reliability Engineering (ISSRE)*, pages 286–296. IEEE. DOI: 10.1109/ISSRE55969.2022.00036.

Ahmadilivani, M. H., Taheri, M., Raik, J., Daneshtalab, M., and Jenihhin, M. (2024). A systematic literature review on hardware reliability assessment methods for deep neural networks. *ACM Computing Surveys*, 56(6):1–39. DOI: 10.1145/3638242.

Arlat, J., Aguera, M., Amat, L., Crouzet, Y., Fabre, J.-C., Laprie, J.-C., Martins, E., and Powell, D. (1990). Fault injection for dependability validation: A methodology and some applications. *IEEE Transactions on Software Engineering*, 16(2):166–182. DOI: 10.1109/32.44380.

Bosio, A., Bernardi, P., Ruospo, A., and Sanchez, E. (2019). A reliability analysis of a deep neural network. In *2019 IEEE Latin American Test Symposium (LATS)*, pages 1–6. IEEE. DOI: 10.1109/LATW.2019.8704548.

Breier, J. and Hou, X. (2022). How practical are fault injection attacks, really? *IEEE Access*, 10:113122–113130. DOI: 10.1109/ACCESS.2022.3217212.

Chen, Z., Narayanan, N., Fang, B., Li, G., Pattabiraman, K., and DeBardeleben, N. (2020). TensorFI: A flexible fault injection framework for TensorFlow applications. In *2020 IEEE 31st International Symposium on Software Reliability Engineering (ISSRE)*, pages 426–435. IEEE. DOI: 10.1109/ISSRE5003.2020.00047.

Dong, J. *et al*. (2023). One-bit flip is all you need: When bit-flip attack meets model training. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*. IEEE. DOI: 10.1109/ICCV51070.2023.00432.

Global Desk (2025). Nvidia chips hacked, fall victim to rowhammer bit-flip attacks; here's how to secure the AI GPUs. *The Economic Times*. Available at:`https://economictimes.indiatimes.com/news/international/us/`

nvidia-chips-hacked-fall-victim-to-rowhammer-bit-flip-attacks/articleshow/122528758.cms Accessed: 2025-07-26.

Göğebakan, A. B., Magliano, E., Carpegna, A., Ruospo, A., Savino, A., and Di Carlo, S. (2024). SpikingJET: Enhancing fault injection for fully and convolutional spiking neural networks. *IEEE*. DOI: 10.1109/IOLTS60994.2024.10616060.

Govarini, G., Ruospo, A., and Sanchez, E. (2023). A fast reliability analysis of image segmentation neural networks exploiting statistical fault injections. In *2023 IEEE 24th Latin American Test Symposium (LATS)*, pages 1–6. IEEE. DOI: 10.1109/LATS58125.2023.10154488.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778. IEEE. DOI: 10.1109/CVPR.2016.90.

Hector, K., Moëllic, P.-A., Dumont, M., and Dutertre, J.-M. (2022). A closer look at evaluating the bit-flip attack against deep neural networks. In *2022 IEEE 28th International Symposium on On-Line Testing and Robust System Design (IOLTS)*. IEEE. DOI: 10.1109/IOLTS56730.2022.9897693.

Huang, H., Liu, C., Xue, X., Liu, B., Li, H., and Li, X. (2024). MRFI: An open-source multiresolution fault injection framework for neural network processing. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 32(7):1325–1335. DOI: 10.1109/TVLSI.2024.3384404.

Huang, Y., Siegel, P. H., and Jiang, A. (2020). Functional error correction for robust neural networks. *IEEE Journal on Selected Areas in Information Theory*, 1(2):267 – 276. DOI: 10.1109/JSAIT.2020.2991430.

Kim, Y., Daly, R., Kim, J. S., Fallin, C., Lee, J. H., Lee, D., Wilkerson, C., Lai, K., and Mutlu, O. (2014). Flipping bits in memory without accessing them: An experimental study of DRAM disturbance errors. In *ACM SIGARCH Computer Architecture News, Volume 42, Issue 3*, pages 361–372. DOI: 10.1145/2678373.2665726.

Lee, N., Ajanthan, T., and Torr, P. H. S. (2019). SNIP: Single-shot network pruning based on connection sensitivity. In *7th International Conference on Learning Representations (ICLR)*. OpenReview.net. Available at:https://openreview.net/forum?id=B1VZqjAcYX.

Leveugle, R., Calvez, A., Maistri, P., and Vanhauwaert, P. (2009). Statistical fault injection: Quantified error and confidence. In *2009 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 502–506. IEEE. DOI: 10.1109/DATE.2009.5090716.

Lin, C. S., Qu, J., and Saileshwar, G. (2025). GPUHammer: Rowhammer attacks on GPU memories are practical. *arXiv preprint*. DOI: 10.48550/arXiv.2507.08166.

Liu, Y., Wei, L., Luo, B., and Xu, Q. (2017). Fault injection attack on deep neural network. In *2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 131–138. IEEE. DOI: 10.1109/ICCAD.2017.8203770.

Mahmoud, A., Aggarwal, N., Nobbe, A., Sanchez Vicarte, J. R., Adve, S. V., Fletcher, C. W., Frosio, I., and Hari, S. K. S. (2020). PyTorchFI: A runtime perturbation tool for DNNs. In *2020 50th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W)*, pages 25–31. IEEE. DOI: 10.1109/DSN-W50199.2020.00014.

Qian, C., Zhang, M., Nie, Y., Lu, S., and Cao, H. (2023). A survey of bit-flip attacks on deep neural network and corresponding defense methods. *Electronics*, 12(4):853. DOI: 10.3390/electronics12040853.

Rakin, A. S., He, Z., and Fan, D. (2019). Bit-flip attack: Crushing neural network with progressive bit search. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 1211–1220. IEEE. DOI: 10.1109/ICCV.2019.00130.

Rakin, A. S. *et al*. (2022). T-BFA: Targeted bit-flip adversarial weight attack. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(11):7928–7939. DOI: 10.1109/TPAMI.2021.3112932.

Reagen, B., Gupta, U., Pentecost, L., Whatmough, P., Lee, S. K., Mulholland, N., Brooks, D., and Wei, G.-Y. (2018). ARES: A framework for quantifying the resilience of deep neural networks. In *2018 55th ACM/ESDA/IEEE Design Automation Conference (DAC)*, pages 1–6. IEEE. DOI: 10.1109/DAC.2018.8465834.

Ronaldo, J., de Lima, O. A., and Sousa, N. (2025). MLBFA: A machine learning-guided framework for efficient bit-flip attacks on deep neural networks. In *2025 XV Symposium on Computing Systems Engineering (SBESC)*, pages 1–6. IEEE. DOI: 10.1109/SBESC68008.2025.11288903.

Ruospo, A., Bosio, A., Ianne, A., and Sanchez, E. (2020). Evaluating convolutional neural networks reliability depending on their data representation. In *2020 23rd Euromicro Conference on Digital System Design (DSD)*, pages 672–679. IEEE. DOI: 10.1109/DSD51259.2020.00109.

Ruospo, A., Reorda, M. S., Mariani, R., and Sanchez, E. (2025). An effective iterative statistical fault injection methodology for deep neural networks. *IEEE Transactions on Computers*, pages 2431 – 2444. DOI: 10.1109/TC.2025.3566863.

Ruospo, A., Sanchez, E., Luza, L. M., Dilillo, L., Traiola, M., and Bosio, A. (2023a). A survey on deep learning resilience assessment methodologies. *Computer*, 56(2):57–66. DOI: 10.1109/MC.2022.3217841.

Ruospo, A. *et al*. (2021). Pros and cons of fault injection approaches for the reliability assessment of deep neural networks. In *2021 IEEE 22nd Latin American Test Symposium (LATS)*, pages 1–5. IEEE. DOI: 10.1109/LATS53581.2021.9651807.

Ruospo, A. *et al*. (2023b). Assessing convolutional neural networks reliability through statistical fault injections. In *2023 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE. DOI: 10.23919/DATE56975.2023.10136998.

Sabbagh, M., Gongye, C., Fei, Y., and Wang, Y. (2019). Evaluating fault resiliency of compressed deep neural networks. In *2019 IEEE International Conference on Embedded Software and Systems (ICESS)*, pages 1–7. IEEE. DOI: 10.1109/ICESS.2019.8782505.

Syed, R. T., Ulbricht, M., Piotrowski, K., and Krstic, M. (2023). A survey on fault-tolerant methodologies for deep neural networks. *Pomiary Automatyka Robotyka*,

27(2):89–98. DOI: 10.14313/PAR_248/89.

Tanaka, H., Kunin, D., Yamins, D. L. K., and Ganguli, S. (2020). Pruning neural networks without any data by iteratively conserving synaptic flow. In *Advances in Neural Information Processing Systems*, volume 33. Curran Associates, Inc. Available at:`https://proceedings.neurips.cc/paper/2020/hash/46a4378f835dc8040c8057beb6a2da52-Abstract.html`.

Torres-Huitzil, C. and Girau, B. (2017). Fault and error tolerance in neural networks: A review. *IEEE Access*, 5:17322–17341. DOI: 10.1109/ACCESS.2017.2742698.

Traiola, M., Kritikakou, A., and Sentieys, O. (2023a). harDNNing: A machine-learning-based framework for fault tolerance assessment and protection of DNNs. In *2023 IEEE European Test Symposium (ETS)*, pages 1–6. IEEE. DOI: 10.1109/ETS56758.2023.10174178.

Traiola, M., Kritikakou, A., and Sentieys, O. (2023b). A machine-learning-guided framework for fault-tolerant DNNs. In *2023 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 1–2. IEEE. DOI: 10.23919/DATE56975.2023.10137033.

Yu, G., Tan, G., Huang, H., Zhang, Z., Chen, P., Natella, R., Zheng, Z., and Lyu, M. R. (2025). A survey on failure analysis and fault injection in AI systems. *ACM Transactions on Software Engineering and Methodology*, 35(1):1–43. DOI: 10.1145/3732777.

Zhang, Y., Itsuji, H., Uezono, T., Toba, T., and Hashimoto, M. (2022). Estimating vulnerability of all model parameters in DNN with a small number of fault injections. In *2022 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE. DOI: 10.1587/transfun.2022VLP0004.

Zhang, Y., Itsuji, H., Uezono, T., Toba, T., and Hashimoto, M. (2023). Vulnerability estimation of DNN model parameters with few fault injections. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, E106-A(3):523–530. DOI: 10.1587/transfun.2022VLP0004.

Zheng, Y., Feng, Z., Hu, Z., and Pei, K. (2021). MindFI: A fault injection tool for reliability assessment of MindSpore applications. In *2021 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)*, pages 235–238. IEEE. DOI: 10.1109/ISSREW53611.2021.00068.